반도체 불량 AI 자동 검출을 위한 딥러닝 프로젝트

Presentation for Deep Learning Project

Data Universe 박주경, 김성호, 김용훈, 윤예은, 이승주

프로젝트 팀 구성 및 역할

Data Universe

박주경 : 총괄, 모델링, 데이터 증식 및 실험

김성호 : 모델링, 데이터 증식 및 실험

김용훈 : 데이터 증식 및 실험

윤예은 : PPT, 웹구현(Flask)

이승주 : 모델링, 데이터 증식 및 실험

목차

01 프로젝트 개요

02 프로젝트 수행 절차 및 방법

03 프로젝트 수행 결과

04 자체 평가 의견

05 참고자료

01 프로젝트 개요 두 가지 문제

01 Problem of limited data

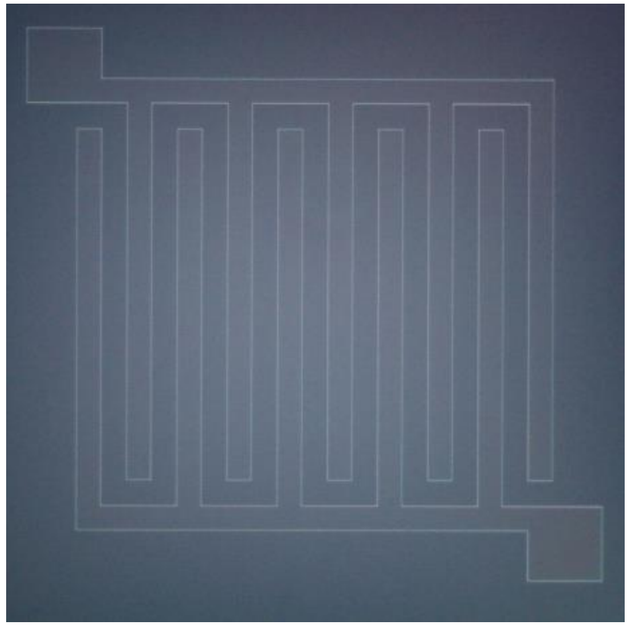
적은 양의 데이터

02 Problem of small object detection

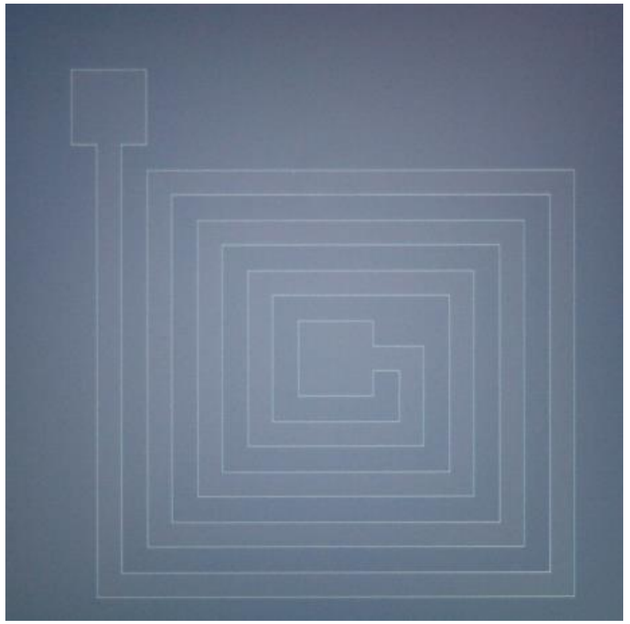
작은 객체 탐지의 어려움

01 프로젝트 개요 적은 양의 데이터

패턴	총 개수	Defect type
패턴 1	136장	9종류
패턴 2	201장	9종류
총개수	337장	



정상 pattern 1



정상 pattern 2

337장 - **23장**(defect 없는 이미지) = **314장**

01 프로젝트 개요 작은 객체 탐지의 문제

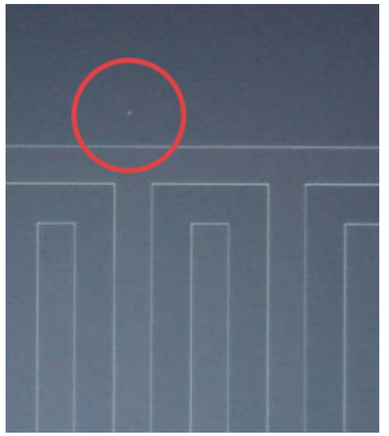
Detection results on MS COCO test-dev set. Bold fonts indicate the best performance(CVPR 2020)

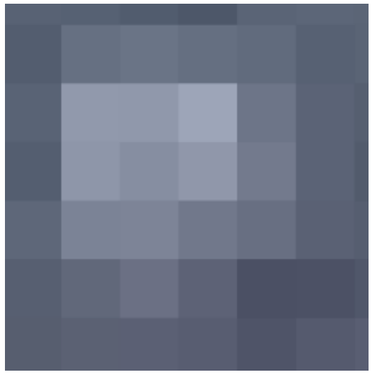
Method	Data	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
anchor-based two-stage:								
MLKP [58]	trainval35	ResNet-101	28.6	52.4	31.6	10.8	33.4	45.1
R-FCN [9]	trainval	ResNet-101	29.9	51.9	-	10.8	32.8	45.0
CoupleNet [74]	trainval	ResNet-101	34.4	54.8	37.2	13.4	38.1	50.8
TDM [53]	trainval	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
Hu et al. [18]	trainval35k	ResNet-101	39.0	58.6	42.9	-	-	-
DeepRegionlets [64]	trainval35k	ResNet-101	39.3	59.8	-	21.7	43.7	50.9
FitnessNMS [57]	trainval	DeNet-101	39.5	58.0	42.6	18.9	43.5	54.1
Gu et al. [15]	trainval35k	ResNet-101	39.9	63.1	43.1	22.2	43.4	51.6
DetNet [31]	trainval35k	DetNet-59	40.3	62.1	43.8	23.6	42.6	50.0
umd_det [3]	trainval	ResNet-101	40.8	62.4	44.9	23.0	43.4	53.2
SOD-MTGAN [1]	trainval35k	ResNet-101	41.4	63.2	45.4	24.7	44.2	52.6
G-RMI [19]	trainval35k	Ensemble of Five Models	41.6	61.9	45.4	23.9	43.5	54.9
C-Mask RCNN [7]	trainval35k	ResNet-101	42.0	62.9	46.4	23.4	44.7	53.8
Cascade R-CNN [5]	trainval35k	ResNet-101	42.8	62.1	46.3	23.7	45.5	55.2
Revisiting RCNN [8]	trainval35k	ResNet-101+ResNet-152	43.1	66.1	47.3	25.8	45.9	55.3
SNIP [54]	trainval35k	DPN-98	45.7	67.3	51.1	29.3	48.8	57.1
anchor-based one-stage:								
YOLOv2 [46]	trainval35k	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5
SSD512* [36]	trainval35k	VGG-16	28.8	48.5	30.3	10.9	31.8	43.5
STDN513 [69]	trainval	DenseNet-169	31.8	51.0	33.6	14.4	36.1	43.4
DES512 [68]	trainval35k	VGG-16	32.8	53.2	34.5	13.9	36.2	47.5
DSSD513 [12]	trainval35k	ResNet-101	33.2	53.3	35.2	13.0	35.4	51.1
RFB512-E [35]	trainval35k	VGG-16	34.4	55.7	36.4	17.6	37.0	47.6
DFP512 [22]	trainval	ResNet-101	34.6	54.3	37.3	-	-	-
PFPNet-R512 [21]	trainval35k	VGG-16	35.2	57.6	37.9	18.7	38.6	45.9
RefineDet512 [66]	trainval35k	ResNet-101	36.4	57.5	39.5	16.6	39.9	51.4
RetinaNet [33]	trainval35k	ResNet-101	39.1	59.1	42.3	21.8	42.7	50.2
anchor-free keypoint-based:								
ExtremeNet [71]	trainval35k	Hourglass-104	40.2	55.5	43.2	20.4	43.2	53.1
CornerNet [26]	trainval35k	Hourglass-104	40.5	56.5	43.1	19.4	42.7	53.9
CenterNet-HG [70]	trainval35k	Hourglass-104	42.1	61.1	45.9	24.1	45.5	52.8
Grid R-CNN [39]	trainval35k	ResNeXt-101	43.2	63.0	46.6	25.1	46.5	55.2
CornerNet-Lite [27]	trainval35k	Hourglass-54	43.2	-	-	24.4	44.6	57.3
CenterNet [11]	trainval35k	Hourglass-104	44.9	62.4	48.1	25.6	47.4	57.4
RepPoints [65]	trainval35k	ResNet-101-DCN	45.0	66.1	49.0	26.6	48.6	57.5
anchor-free center-based:								
GA-RPN [59]	trainval35k	ResNet-50	39.8	59.2	43.5	21.8	42.6	50.7
FoveaBox [23]	trainval35k	ResNeXt-101	42.1	61.9	45.2	24.9	46.8	55.6
FSAF [72]	trainval35k	ResNeXt-64x4d-101	42.9	63.8	46.3	26.6	46.2	52.7
FCOS [56]	trainval35k	ResNeXt-64x4d-101	43.2	62.8	46.6	26.5	46.2	53.3
Ours:								
ATSS	trainval35k	ResNet-101	43.6	62.1	47.4	26.1	47.0	53.6
ATSS	trainval35k	ResNeXt-32x8d-101	45.1	63.9	49.1	27.9	48.2	54.6
ATSS	trainval35k	ResNeXt-64x4d-101	45.6	64.6	49.7	28.5	48.9	55.6
ATSS	trainval35k	ResNet-101-DCN	46.3	64.7	50.4	27.7	49.8	58.4
ATSS	trainval35k	ResNeXt-32x8d-101-DCN	47.7	66.6	52.1	29.3	50.8	59.7
ATSS	trainval35k	ResNeXt-64x4d-101-DCN	47.7	66.5	51.9	29.7	50.8	59.4
ATSS (Multi-scale testing)	trainval35k	ResNeXt-32x8d-101-DCN	50.6	68.6	56.1	**33.6**	**52.9**	62.2
ATSS (Multi-scale testing)	trainval35k	ResNeXt-64x4d-101-DCN	**50.7**	**68.9**	**56.3**	33.2	**52.9**	**62.4**

End-to-End Oject Detection with Transformers (ECCV2020)

Model	GFLOPS/FPS	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	**47.8**	**27.2**	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	**44.9**	**64.7**	47.7	23.7	**49.5**	**62.3**

APs - 32픽셀 미만
APm - 32 ~ 96 픽셀 미만
APL - 96픽셀 이상





(4 x 4)

02 프로젝트 수행 절차 및 방법 적은 데이터 셋의 문제

적은 데이터 셋을 다루는

유효한 3가지 방식

Data Augmentation
데이터 증식

Transfer Learning
전이 학습

Fine Tuning
미세 조정

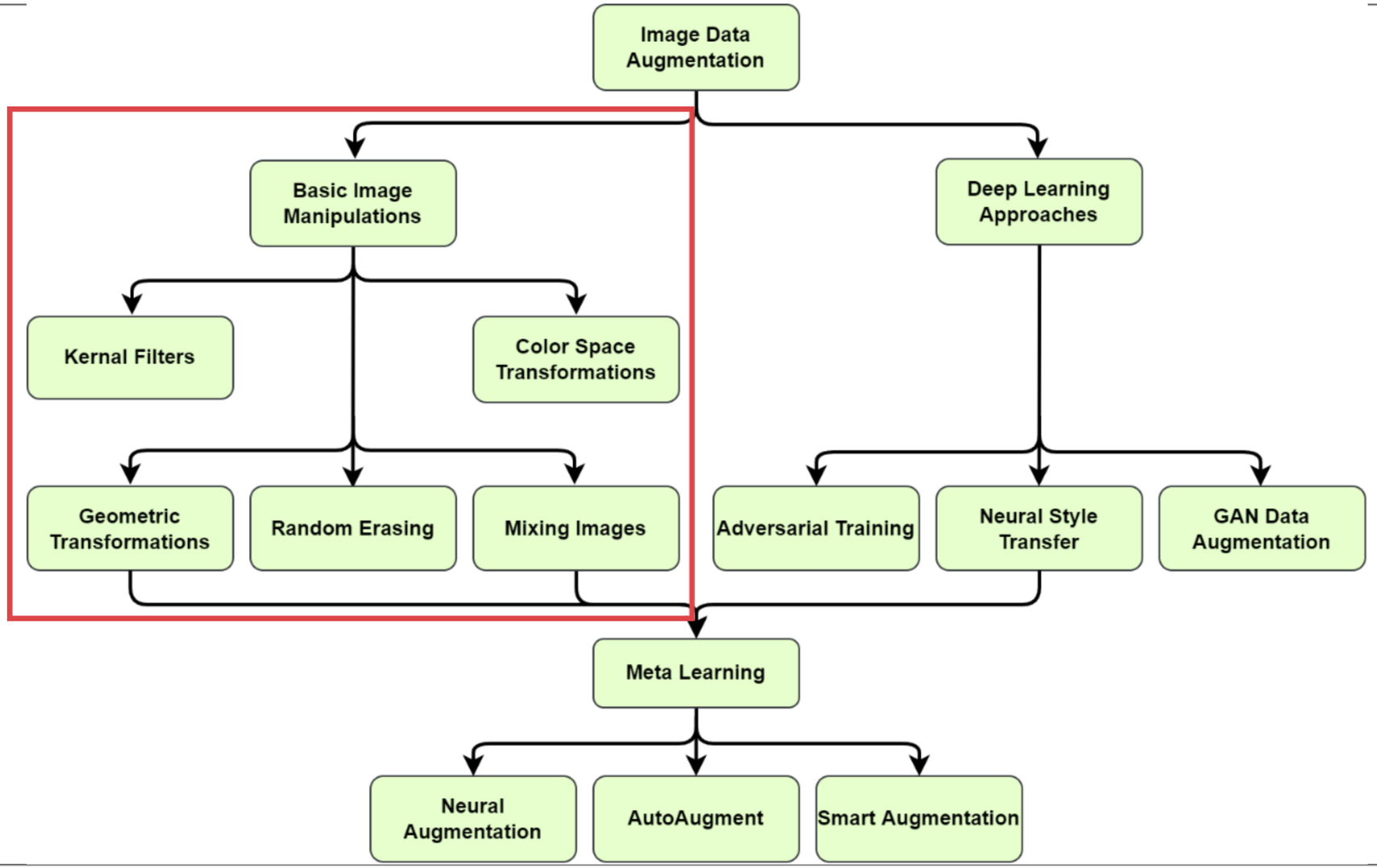
<Overfitting on Model Traning>

- Advanced architectures
- Functional solutions
- One-shot / Zero-shot Learning
- Data Augmentation

<yolov8>

pretrained on the COCO dataset

02 프로젝트 수행 절차 및 방법 적은 데이터 셋의 문제

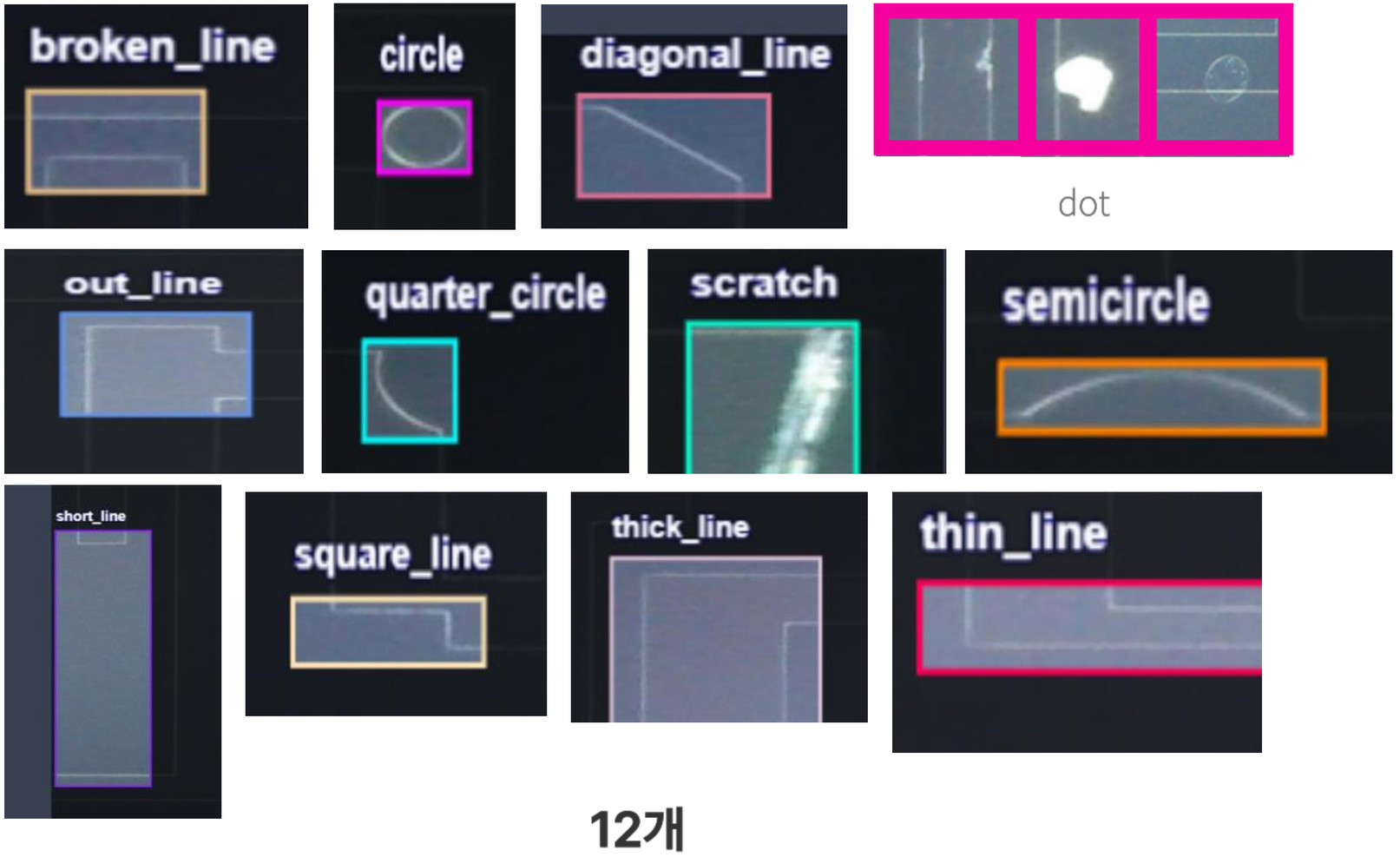


02 프로젝트 수행 절차 및 방법 적은 데이터 셋의 문제

Class

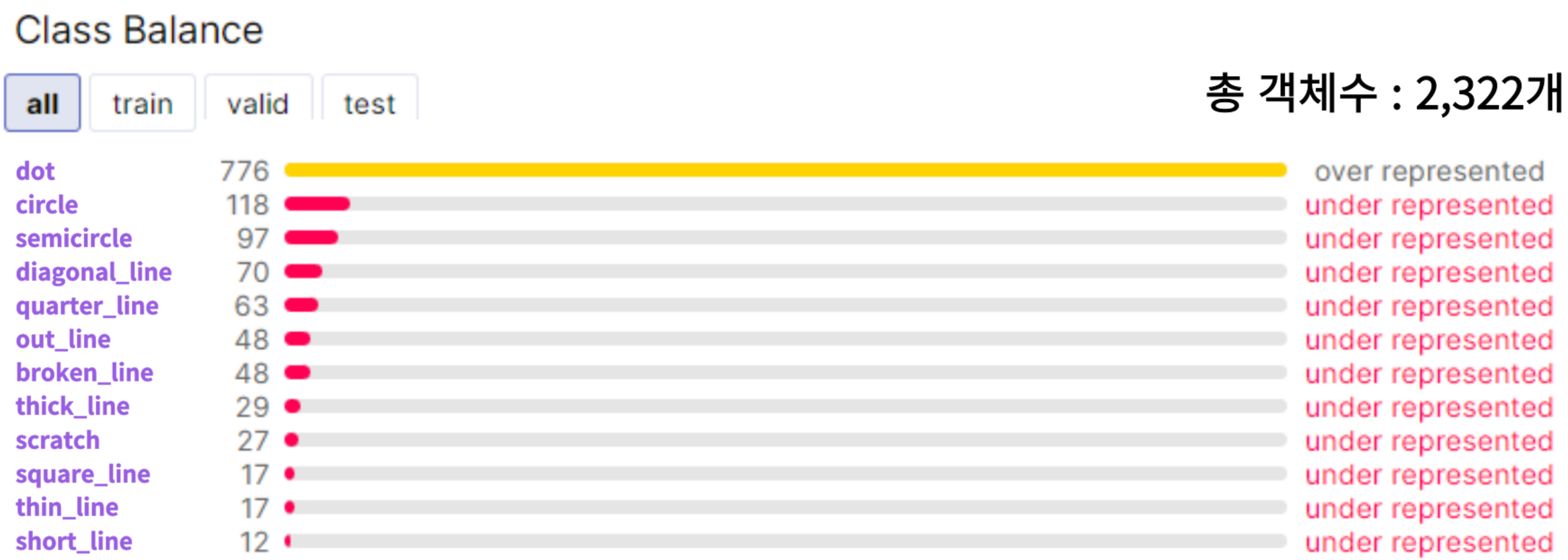
Defect 유형

9개

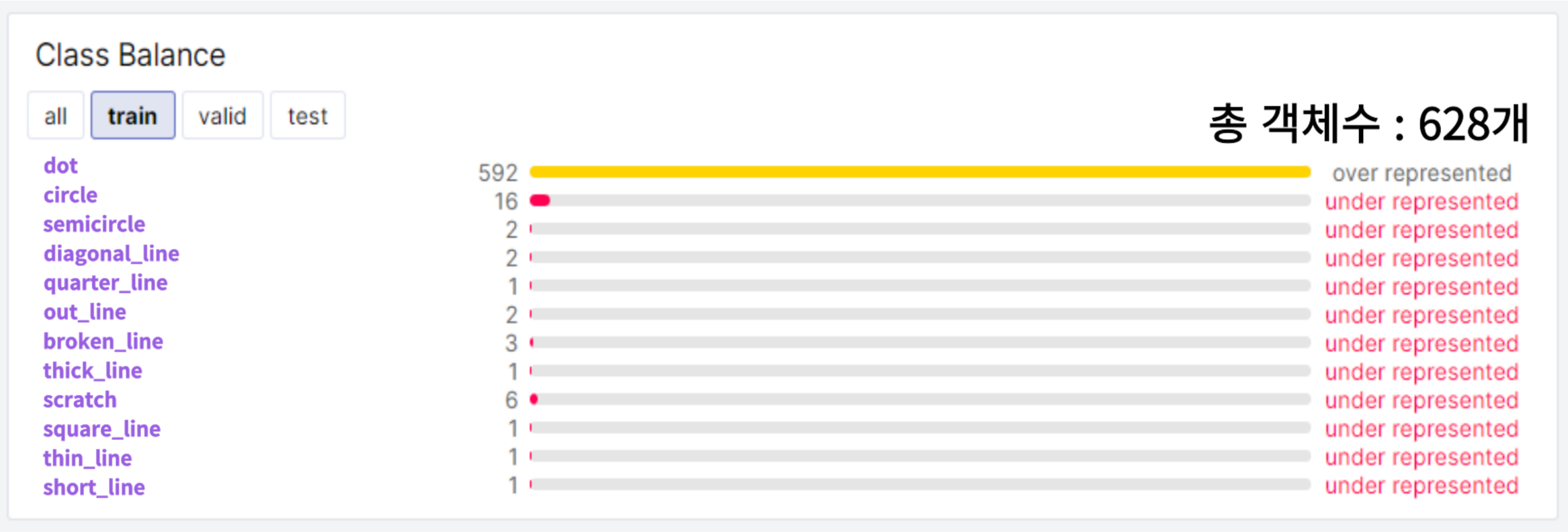


12개

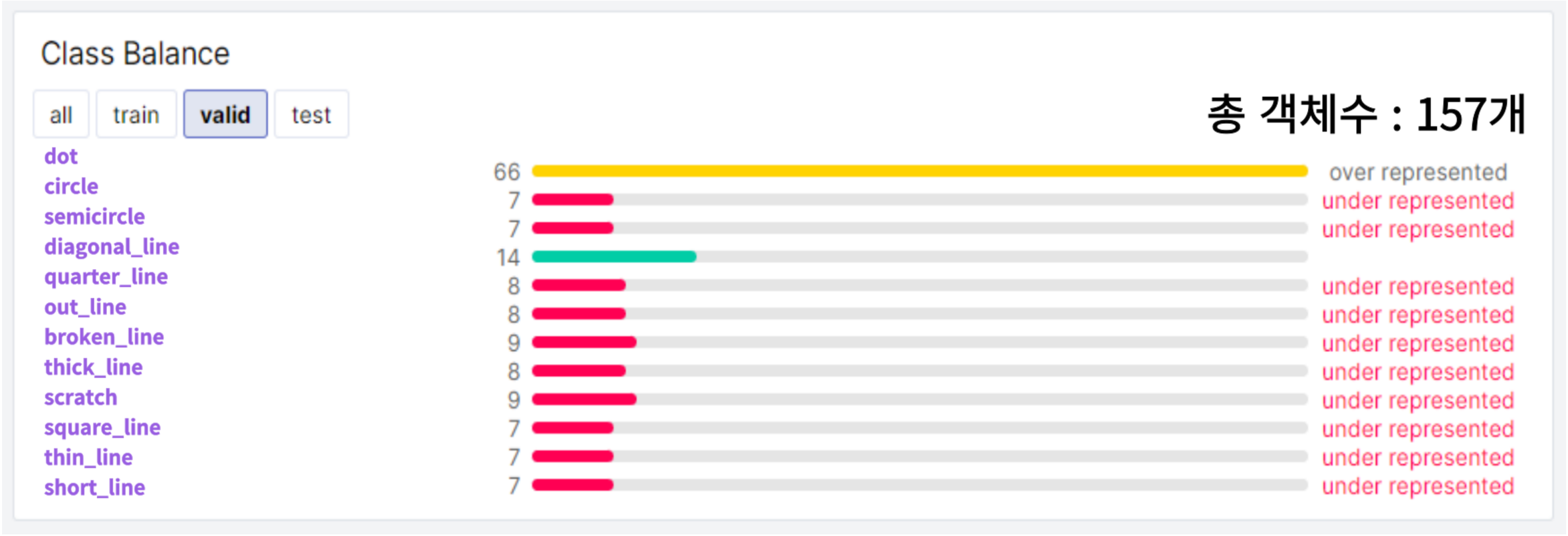
02 프로젝트 수행 절차 및 방법 적은 데이터 셋의 문제



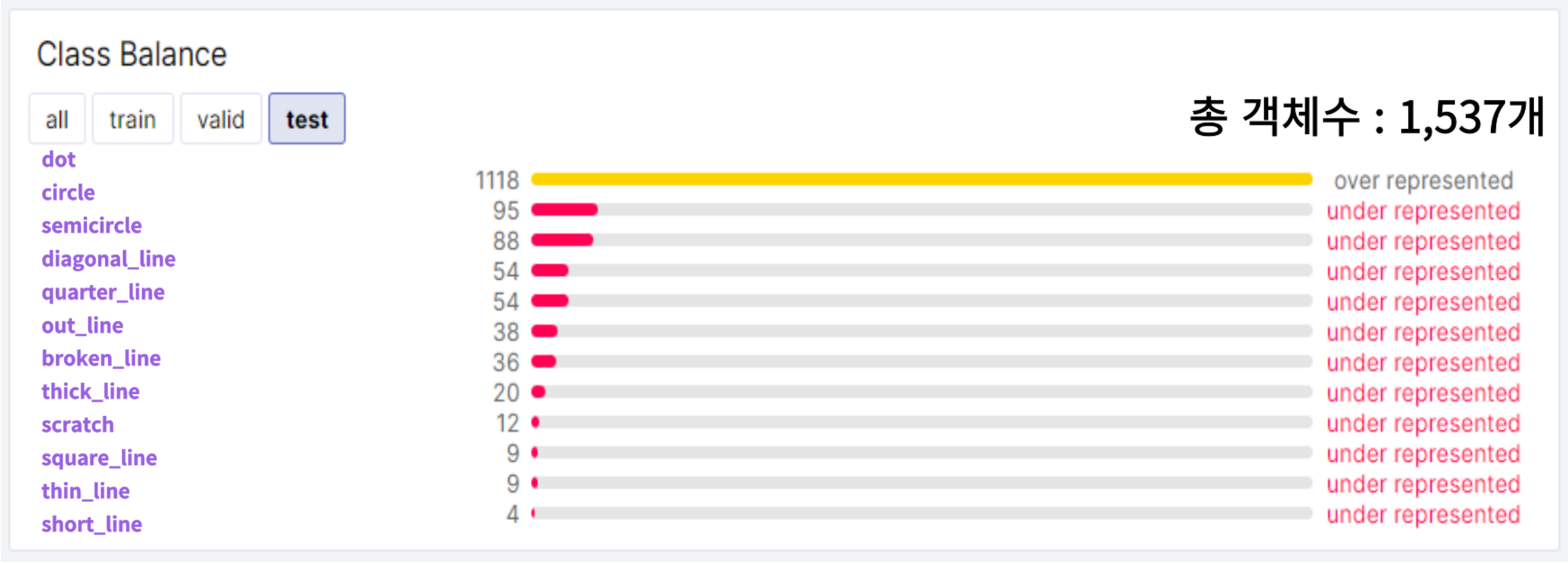
02 프로젝트 수행 절차 및 방법 적은 데이터 셋의 문제



02 프로젝트 수행 절차 및 방법 적은 데이터 셋의 문제



02 프로젝트 수행 절차 및 방법 적은 데이터 셋의 문제



02 프로젝트 수행 절차 및 방법 작은 객체 탐지의 문제

증식 유형 총 99가지 in albumentations & roboflow

AdvancedBlur
Affine
Augmix
BBoxSafeRandomCrop
Blur
Brightness
CenterCrop
CLAHE
ChannelDroout
ChannelShuffle
CoarseDropout
Color Jitter
Crop
CropAndPad
CropNonEmptyMaskIfExists
Cutmix
Cutout
Defocus
Downscale
ElasticTransform
Emboss
Equalize
Exposure

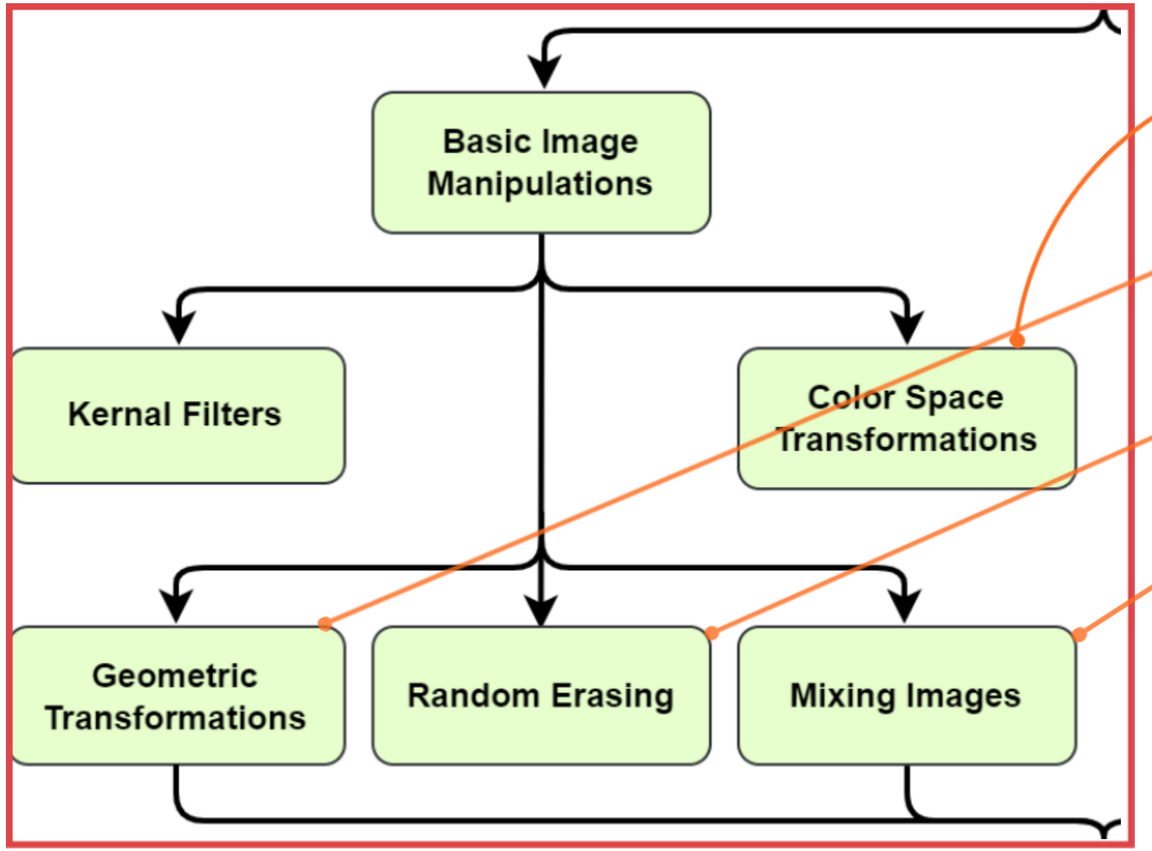
FancyPCA
FDA
Flip
FromFloat
GaussNoise
GaussianBlur
GrayScale
GlassBlur
GridDistortion
GridDroupout
HistogramMatching
HorizontalFlip
Hue
HueSaturationValue
ISONoise
ImageCompression
InvertImg
Lambda
LongestMaxSize
MaskDroupout
MedianBlur
Mixup
MotionBlur

Mosaic
MultiplicativeNoise
Noise
NoOp
Normalize
OpticalDistortion
PadIfNeeded
Perspective
PiecewiseAffine
PixelDistributionAdaptation
PixelDroupout
Posterize
Puzzle Mix
RGBShift
RandomBrightnessContrast
RandomCrop
RandomCropFromBorders
RandomCropNearBBox
RandomFog
RandomGamma
RandomGravel
RadomGridShuffle
RandomRain

RandomResizedCrop
RandomRotate90
RandomScale
RandomShadow
RandomSizedBBoxSafeCrop
RandomSizedCrop
RandomSnow
RandomSunFlare
RandomToneCurve
Resize
RingingOvershoot
Rotate
SafeRotate
Saturation
Sharpen
Shear
ShiftScaleRotate
SmallestMaxSize
Solarize
Spatter
Superpixels
TemplateTransform
ToFloat

ToGray
ToRGB
ToSepia
Transpose
UnsharpMask
VerticalFlip
ZoomBlur

02 프로젝트 수행 절차 및 방법 작은 객체 탐지의 문제



hue, brightness, saturtion ...

rotation, flip, shear ...

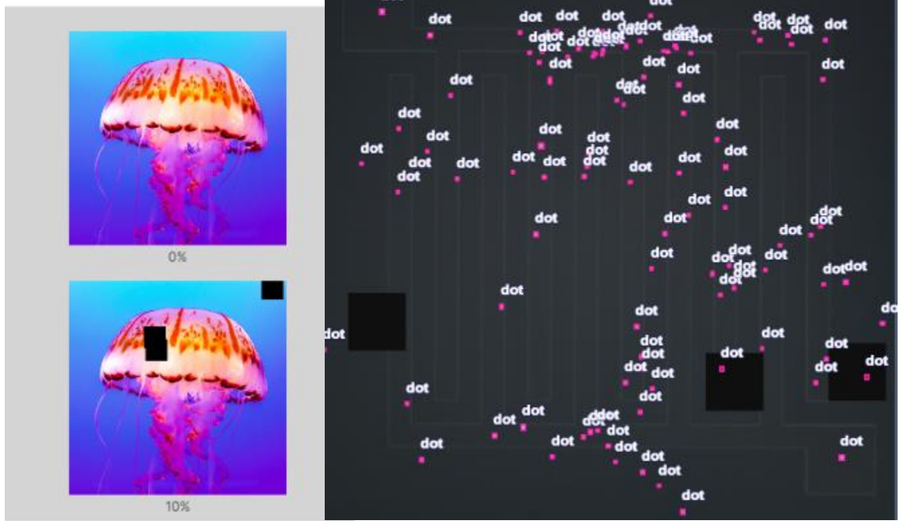
cutout ...

crop, mosaic ...

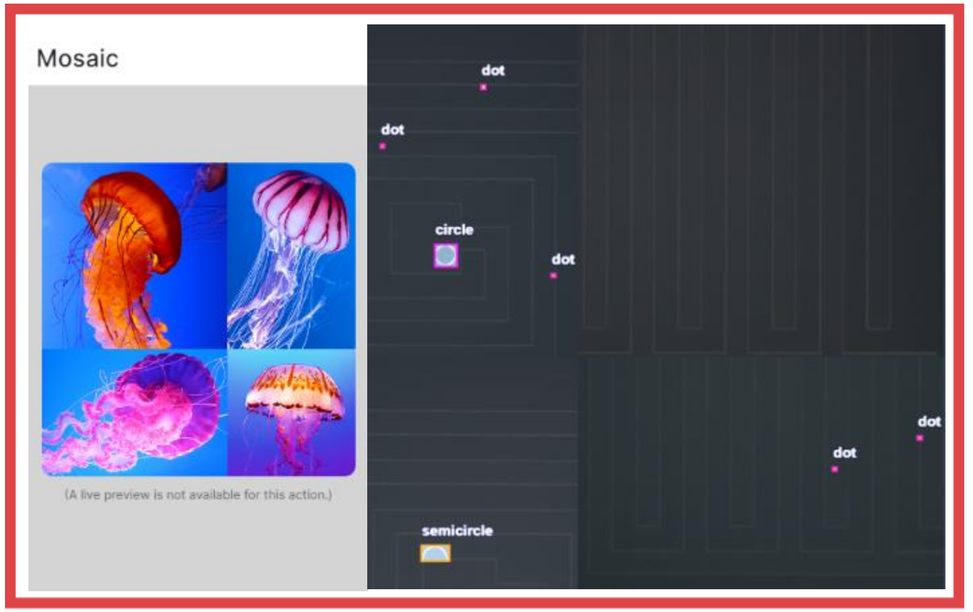
- **task 유형에 따른 다른 증식 방법 적용**
  (detection, segmentation, classification ...)
- **object 유형에 따른 다른 증식 방법 적용**

02 프로젝트 수행 절차 및 방법 작은 객체 탐지의 문제

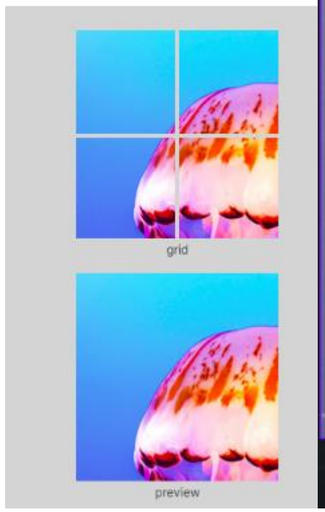
Cutout

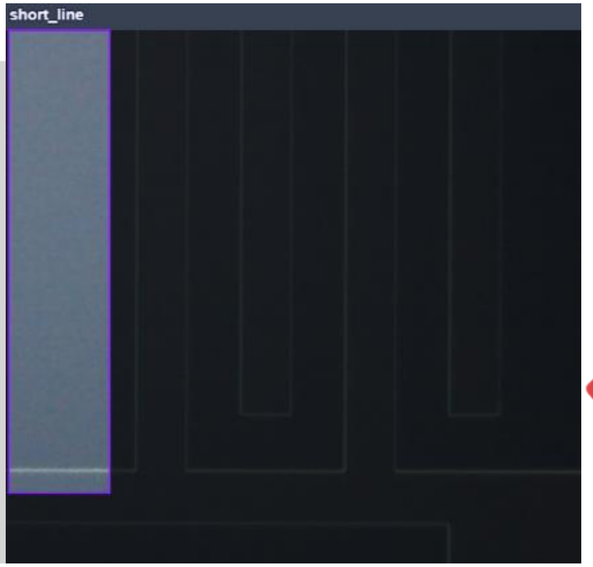


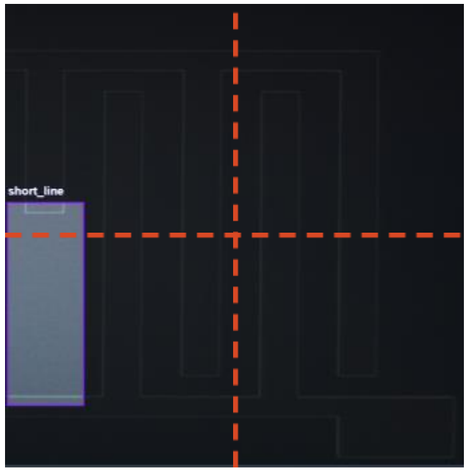
Mosaic



Tile







03 프로젝트 수행 결과

추후에 데이터가 추가되면 주기적으로 훈련 - 일반화, 다양성에 적응

< 적용 전 >

train : 93장 / valid : 344장 **640 px**

Class	Images	Instances	Box(P	R	mAP50	mAP50-95
all	344	1296	0.683	0.404	0.431	0.275
broken_line	344	72	1	0.0511	0.222	0.0966
circle	344	56	0.951	1	0.995	0.856
diagonal_line	344	112	0.647	0.857	0.768	0.446
dot	344	536	0.532	0.262	0.282	0.125
out_line	344	96	0.84	0.383	0.526	0.336
quarter_circle	344	64	0.838	0.25	0.512	0.256
scratch	344	72	0.163	0.5	0.326	0.161
semicircle	344	56	0.552	0.75	0.725	0.565
short_line	344	56	0.642	0.286	0.354	0.226
square_line	344	56	1	0	0	0
thick_line	344	64	0.294	0.328	0.209	0.119
thin_line	344	56	0.741	0.179	0.248	0.109

resize

< 적용 후 >

1280 px

Class	Images	Instances	Box(P	R	mAP50	mAP50-95
all	344	1296	0.623	0.467	0.466	0.29
broken_line	344	72	0.642	0.224	0.32	0.14
circle	344	56	0.97	1	0.995	0.865
diagonal_line	344	112	0.543	0.902	0.851	0.533
dot	344	536	0.447	0.716	0.579	0.259
out_line	344	96	0.77	0.635	0.672	0.357
quarter_circle	344	64	0.159	0.125	0.292	0.169
scratch	344	72	0.113	0.528	0.188	0.0945
semicircle	344	56	0.781	0.804	0.865	0.639
short_line	344	56	0.784	0.232	0.359	0.202
square_line	344	56	0.931	0.143	0.216	0.116
thick_line	344	64	0.34	0.297	0.231	0.0967
thin_line	344	56	1	0	0.022	0.00742

03 프로젝트 수행 결과

추후에 데이터가 추가되면 주기적으로 훈련 - 일반화, 다양성에 적응

train : 93 → 461장 / valid : 344장

< 적용 전 >

Class	mAP50
all	0.466
dot	0.579
scratch	0.188

flip

< 적용 후 >

Class	mAP50
all	0.844
dot	0.698
scratch	0.334

train : 93 → 369장 / valid : 344장

< 적용 전 >

Class	mAP50
all	0.466
dot	0.579
scratch	0.188

rotation

< 적용 후 >

Class	mAP50
all	0.825
dot	0.676
scratch	0.361

train : 93 → 279장 / valid : 344장

< 적용 전 >

Class	mAP50
all	0.466
dot	0.579
scratch	0.188

shear

< 적용 후 >

Class	mAP50
all	0.548
dot	0.61
scratch	0.189

train : 93 → 1912장 / valid : 344장

< 적용 전 >

Class	mAP50
all	0.466
dot	0.579
scratch	0.188

crop (4×4)

< 적용 후 >

Class	mAP50
all	0.736
dot	0.659
scratch	0.493

03 프로젝트 수행 결과

추후에 데이터가 추가되면 주기적으로 훈련 - 일반화, 다양성에 적응

train : 93 → 369장 / valid : 344장

< 적용 전 >

Class	mAP50
all	0.466
dot	0.579
scratch	0.188

normalization

< 적용 후 >

Class	mAP50
all	0.524
dot	0.637
scratch	0.177

train : 93 → 1379장 / valid : 344장

< 적용 전 >

Class	mAP50
all	0.466
dot	0.579
scratch	0.188

mosaic

< 적용 후 >

Class	mAP50
all	0.96
dot	0.738
scratch	0.833

03 프로젝트 수행 결과

추후에 데이터가 추가되면 주기적으로 훈련 - 일반화, 다양성에 적응

train : 93장

Class	mAP50
all	0.431
dot	0.282
scratch	0.326

rotation+flip

train : 737장

Class	mAP50
all	0.861
dot	0.701
scratch	0.358

rotation+flip+crop

train : 2649 장

Class	mAP50
all	0.901
dot	0.724
scratch	0.538

rotation+flip+mosaic

train : 2023장

Class	mAP50
all	0.97
dot	0.751
scratch	0.94

rotation+flip
+mosaic+shear

train : 2209장

Class	mAP50
all	0.958
dot	0.753
scratch	0.868

rotation+flip
+mosaic+crop

train : 3935장

Class	mAP50
all	0.959
dot	0.766
scratch	0.808

rotation+flip+mosaic
+crop+shear

train : 4121장

Class	mAP50
all	0.969
dot	0.799
scratch	0.894

train : 2023장

Class	mAP50
all	0.968
dot	0.825
scratch	0.838

yolov8m

03 프로젝트 수행 결과

추후에 데이터가 추가되면 주기적으로 훈련 - 일반화, 다양성에 적응

train : 93→ 2023장

Class	mAP50
all	0.431
dot	0.282
scratch	0.326

rotation+flip+mosaic

161분 yolov8n

Class	mAP50
all	0.97
dot	0.751
scratch	0.94

184분 yolov8s

Class	mAP50
all	0.967
dot	0.777
scratch	0.886

386분 yolov8m

Class	mAP50
all	0.968
dot	0.825
scratch	0.838

592분 yolov8l

Class	mAP50
all	0.96
dot	0.826
scratch	0.75

모델이 커질 수록

dot은 잘 잡지만 scratch는 떨어짐

03 프로젝트 수행 결과

추후에 데이터가 추가되면 주기적으로 훈련 - 일반화, 다양성에 적응

< 적용 전 >

train : 93장

Class	Images	Instances	Box(P	R	mAP50	mAP50-95
all	344	1296	0.683	0.404	0.431	0.275
broken_line	344	72	1	0.0511	0.222	0.0966
circle	344	56	0.951	1	0.995	0.856
diagonal_line	344	112	0.647	0.857	0.768	0.446
dot	344	536	0.532	0.262	0.282	0.125
out_line	344	96	0.84	0.383	0.526	0.336
quarter_circle	344	64	0.838	0.25	0.512	0.256
scratch	344	72	0.163	0.5	0.326	0.161
semicircle	344	56	0.552	0.75	0.725	0.565
short_line	344	56	0.642	0.286	0.354	0.226
square_line	344	56	1	0	0	0
thick_line	344	64	0.294	0.328	0.209	0.119
thin_line	344	56	0.741	0.179	0.248	0.109

rotation+flip+mosaic

train : 2023장

Class	Images	Instances	Box(P	R	mAP50	mAP50-95
all	344	1296	0.952	0.955	0.968	0.735
broken_line	344	72	0.995	1	0.995	0.676
circle	344	56	0.987	1	0.995	0.9
diagonal_line	344	112	0.996	1	0.995	0.723
dot	344	536	0.809	0.726	0.825	0.42
out_line	344	96	0.995	1	0.995	0.883
quarter_circle	344	64	0.993	1	0.995	0.717
scratch	344	72	0.79	0.736	0.838	0.558
semicircle	344	56	0.99	1	0.995	0.829
short_line	344	56	0.992	1	0.995	0.86
square_line	344	56	0.992	1	0.995	0.587
thick_line	344	64	0.884	1	0.993	0.901
thin_line	344	56	1	1	0.995	0.761

03 프로젝트 수행 결과

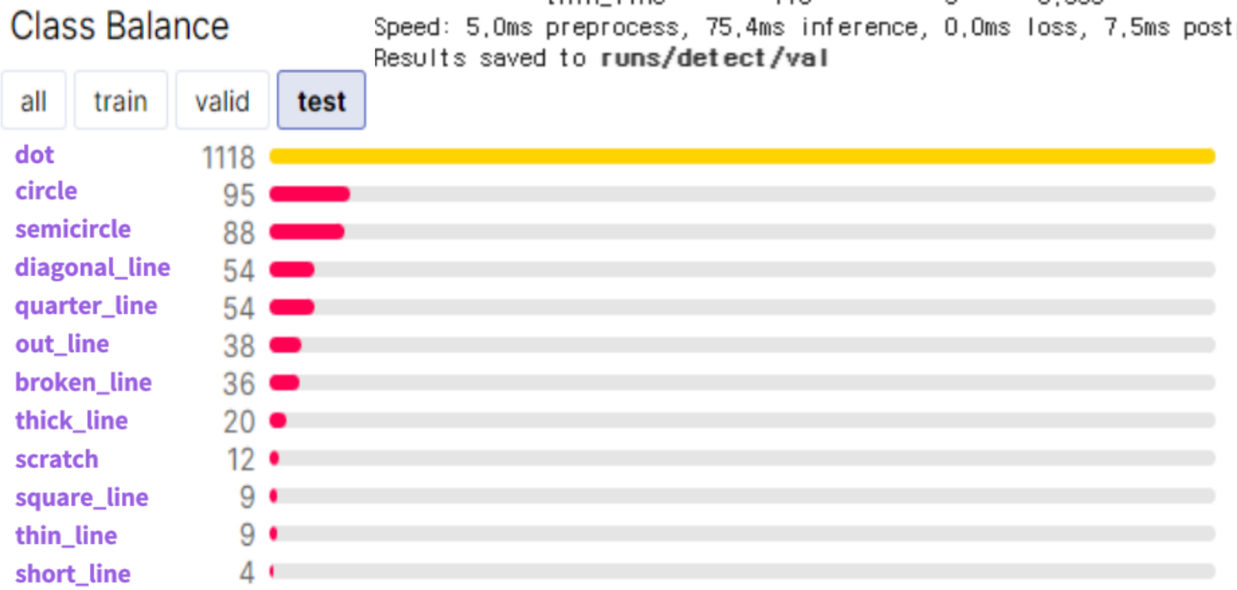
추후에 데이터가 추가되면 주기적으로 훈련 - 일반화, 다양성에 적응

```
all_ap: array([[     0.995,      0.995,      0.995,      0.995,     0.99114,     0.70607,     0.46507,     0.36148,     0.20309,    0.058014],
       [     0.995,      0.995,      0.995,      0.995,      0.995,      0.995,      0.995,      0.995,     0.84913,     0.41435],
       [     0.995,      0.995,     0.98728,     0.93032,     0.90067,     0.80647,     0.64381,     0.50524,      0.4117,    0.072771],
       [    0.82876,     0.81227,     0.73412,     0.65965,     0.57286,     0.44199,     0.26169,     0.15927,    0.057268,   0.0021339],
       [     0.995,      0.995,      0.995,      0.995,      0.995,      0.995,     0.93772,     0.92922,     0.79286,     0.17663],
       [     0.995,      0.995,      0.995,      0.995,      0.995,      0.995,     0.77553,     0.21824,     0.11416,    0.031335],
       [    0.83666,     0.80369,     0.79538,     0.74462,     0.71988,     0.69734,     0.47995,      0.2742,     0.17218,    0.048511],
       [     0.995,      0.995,      0.995,      0.995,      0.995,      0.995,      0.995,     0.93063,     0.39273,    0.078718],
       [     0.995,      0.995,      0.995,      0.995,     0.85278,     0.85278,     0.85278,     0.85278,     0.80674,     0.41568],
       [     0.995,      0.995,     0.98178,      0.8134,     0.76295,     0.58227,     0.37498,     0.24099,     0.13592,    0.017867],
       [    0.99255,     0.99255,     0.99255,     0.99255,     0.99255,     0.99255,     0.99255,     0.95314,     0.88794,     0.26719],
       [     0.995,      0.995,      0.995,      0.995,      0.995,     0.96788,     0.81548,     0.52847,     0.26822,   0.0095343]])
ap: array([    0.67649,     0.92235,     0.72483,       0.453,     0.88064,     0.71093,     0.55724,     0.83671,     0.86135,     0.59002,     0.90561,
0.75646])
ap50: array([     0.995,      0.995,      0.995,     0.82876,      0.995,      0.995,     0.83666,      0.995,      0.995,      0.995,     0.99255,
0.995])
ap_class_index: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
f1: array([    0.9974,     0.99346,     0.99808,     0.76638,     0.99768,     0.99628,     0.76119,     0.99483,     0.99595,     0.99596,     0.93833,
1])
map: 0.7396343559741697
map50: 0.96774713822909
map75: 0.8356119630444551
maps: array([    0.67649,     0.92235,     0.72483,       0.453,     0.88064,     0.71093,     0.55724,     0.83671,     0.86135,     0.59002,     0.90561,
0.75646])
mp: 0.9521330369052013
mr: 0.954999309010503
nc: 12
p: array([    0.99482,     0.98701,     0.99616,     0.81417,     0.99538,     0.99259,     0.78804,     0.98972,     0.99192,     0.99196,     0.88382,
1])
r: array([          1,           1,           1,     0.72388,           1,           1,     0.73611,           1,           1,           1,           1,
1])
```

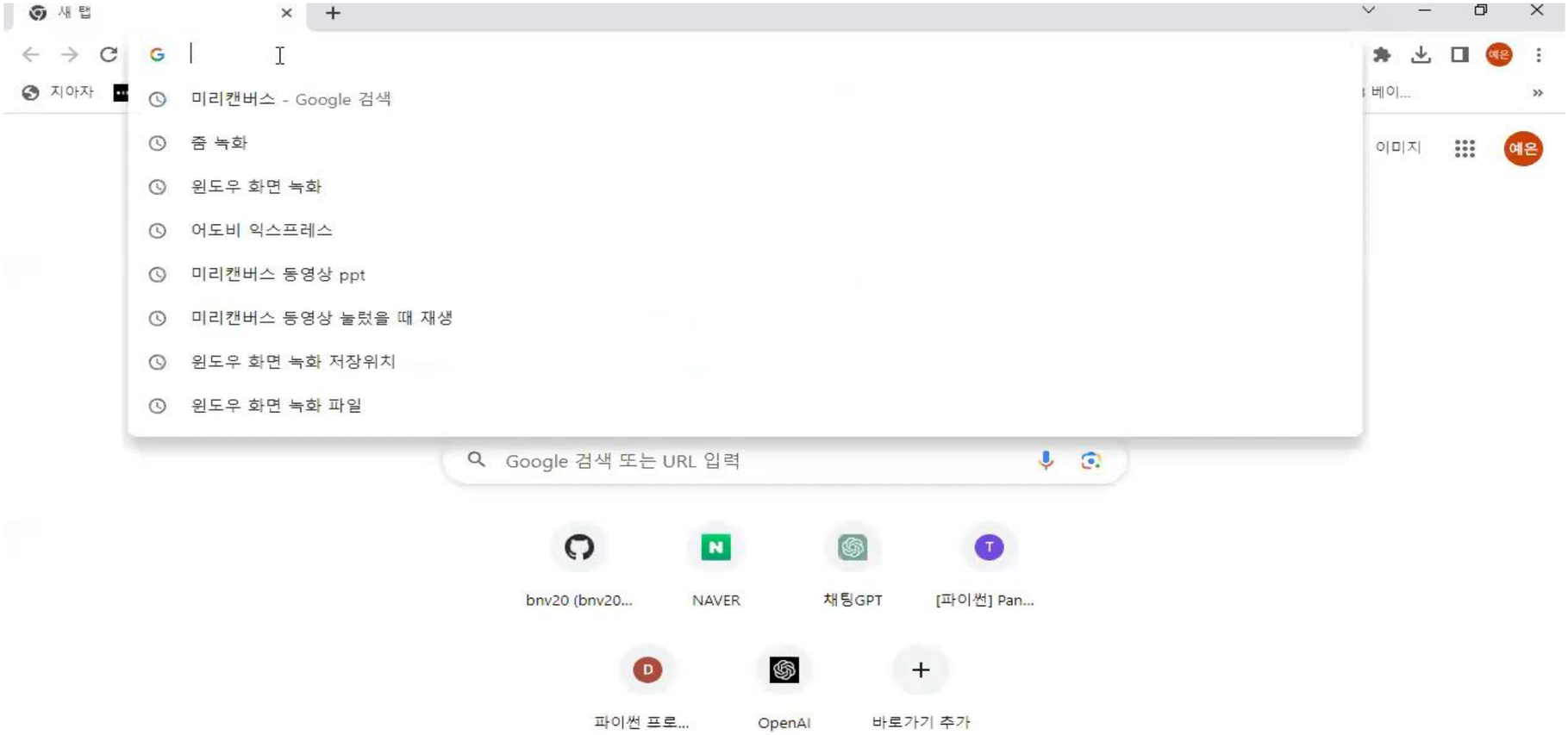
03 프로젝트 수행 결과 추후에 데이터가 추가되면 주기적으로 훈련 - 일반화, 다양성에 적응

predict test set


```
val: New cache created: /content/drive/.shortcut-targets-by-id/1j1XOnE3ISNWuKsNmOKDY3zGs
                   Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                     all        178       1503      0,935      0,939      0,952      0,755
             broken_line        178         36       0,99          1      0,995      0,718
                  circle        178         95      0,991          1      0,995       0,91
            diagonal_line        178         54      0,976          1      0,995      0,804
                     dot        178       1091      0,907      0,607      0,872      0,471
                out_line        178         32      0,992          1      0,995      0,853
           quarter_circle        178         54      0,993          1      0,995      0,773
                 scratch        178         12      0,522      0,667      0,606      0,352
              semicircle        178         88          1      0,994      0,995      0,826
              short_line        178          4      0,913          1      0,995      0,914
             square_line        178          9          1          1      0,995      0,687
              thick_line        178         19      0,978          1      0,995      0,915
               thin_line        178          9      0,959          1      0,995      0,837
Speed: 5.0ms preprocess, 75.4ms inference, 0.0ms loss, 7.5ms postprocess per image
Results saved to runs/detect/val
```



03 프로젝트 수행 결과 웹구현 Flask



04 자체 평가 의견 Conclusion

Data Augmentation은 적은 양의 데이터나 작은 객체 탐지에 **효과적임**

모델 성능에 부정적인 영향을 주지 않는 한, **데이터 증식은 많을수록 좋음**

task, object의 유형이나 크기에 따라 적용해야할 증식 방식이 다름

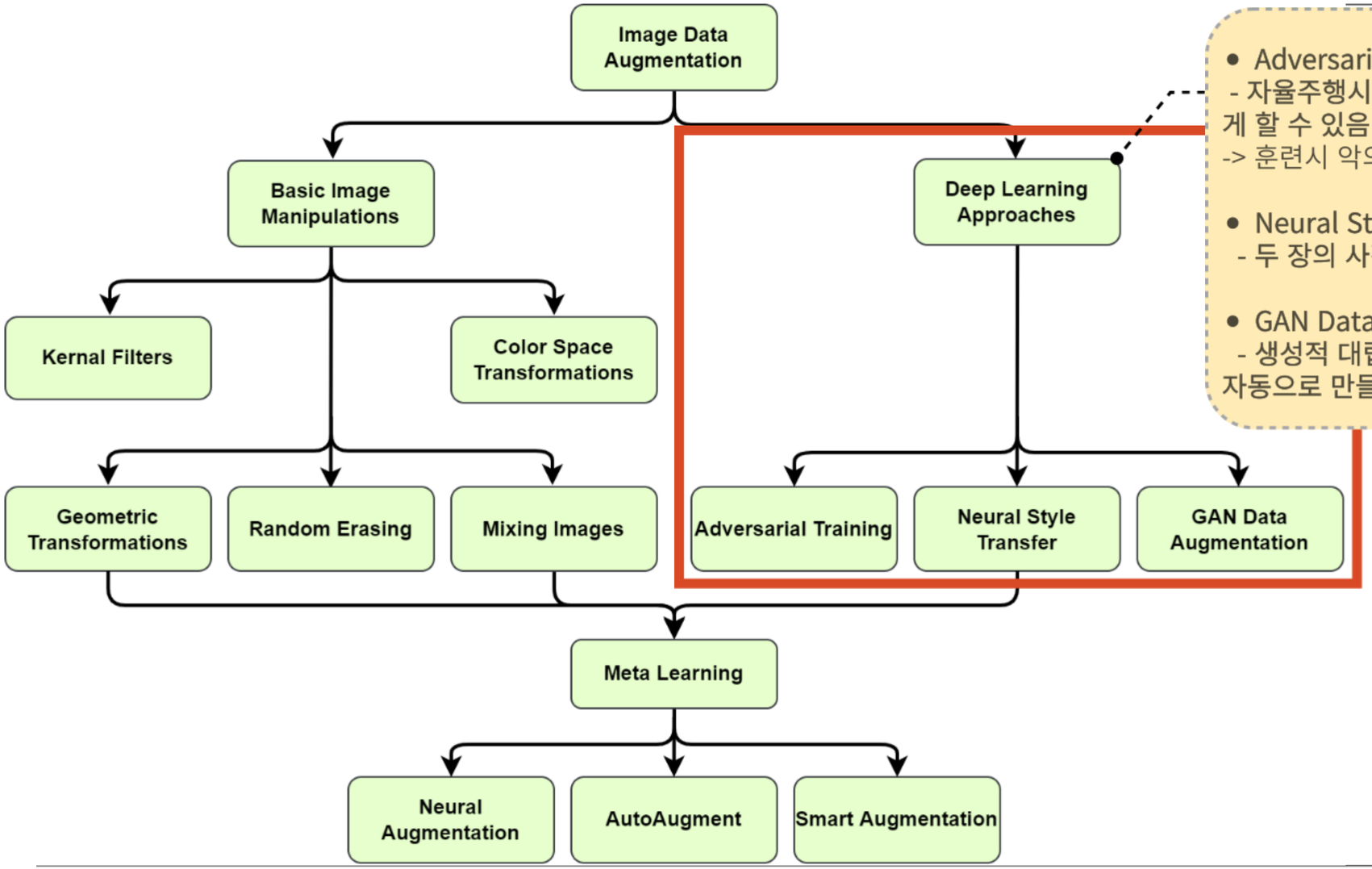
작은 객체 문제에 있어 mosaic이라는 Mixing Image 증식 방식이 상당히 유의미한 효과를 보임

모델 성능이 어느 정도에 이르면, epoch을 늘려도 더 나아지지 않고 과적합으로 흐름 (epoch 100 → 200)
마찬가지로, 모델 성능이 어느 정도에 이르면 **증식을 더 늘려도 큰 모델 성능의 개선은 이루어지지 않음**

데이터의 잡음이나 변동성에서 모델의 강인성을 위해 noise 등의 증식 방법 → **작은 객체와 노이즈 간 모델의 혼란 야기**

이번 프로젝트에서 **cutout 등의 Random Erasing 방식은 작은 객체나 중요한 영역에 대한 정보 손실 발생**

04 자체 평가 의견 Future work

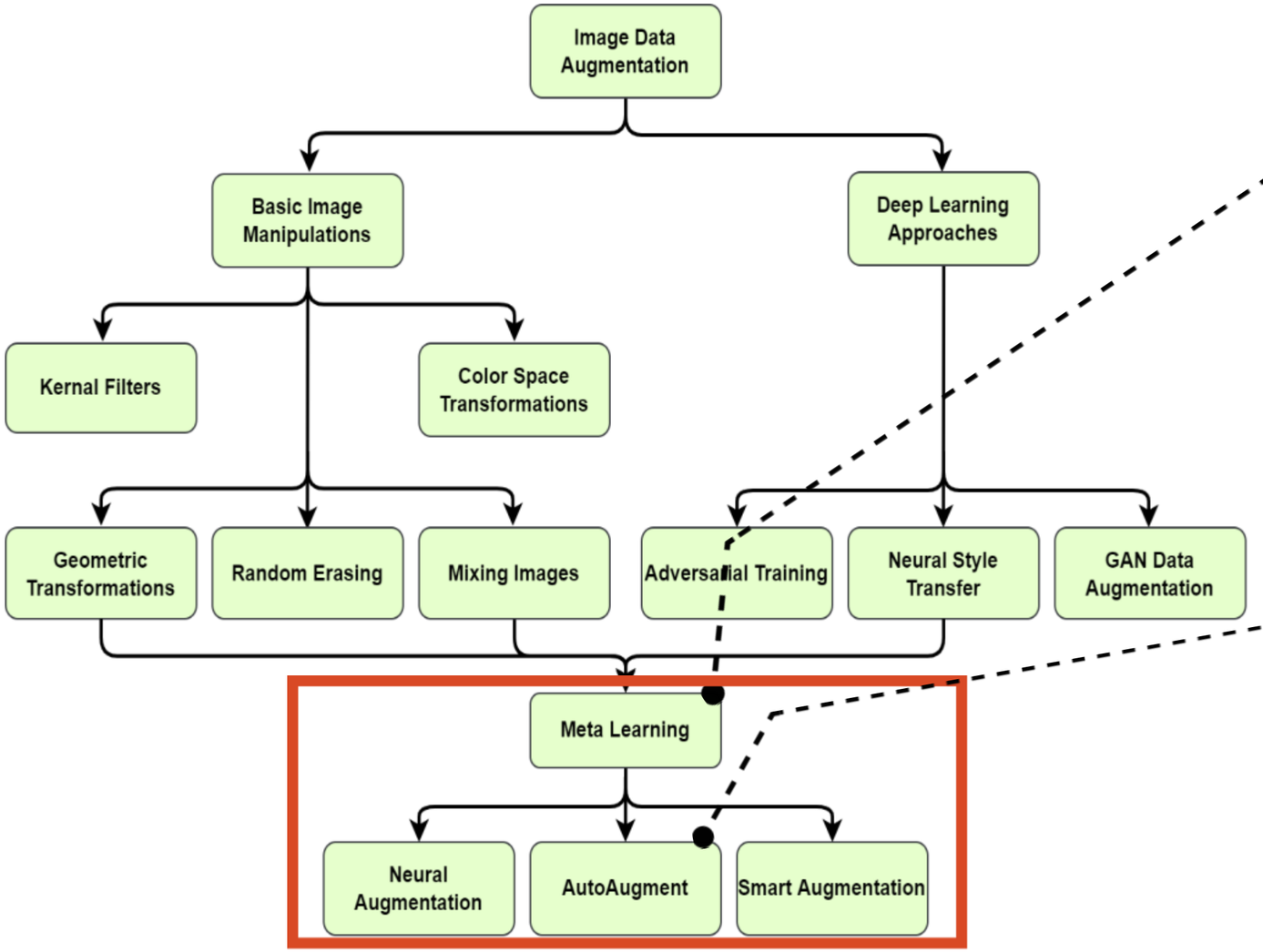


- Adversarial Training (적대적 훈련)
  - 자율주행시 교차로 바닥에 작은 스티커 몇 개를 붙여 오판하게 할 수 있음
  -> 훈련시 악의적 데이터를 도입시킴

- Neural Style Transfer (신경 스타일 전송)
  - 두 장의 사진으로 사진 스타일을 바꾸는데 주로 사용

- GAN Data Augmentation (GAN 데이터 증식)
  - 생성적 대립 신경망, 경쟁을 통해 실제와 가까운 이미지 등을 자동으로 만들어 냄

04 자체 평가 의견 Future work



Meta Learning
: Few-shot Learning 을 위해 제안된 알고리즘

Auto Augment(AA, 2019)

- 자주 사용되는 16가지의 증식 방법들로부터 가장 최적의 조합을 해줌
- 학습이 오래 걸림

Population-Based Augmentation (PBA, 2019)

- FastAutoAugment (FastAA, 2019)
- Bayesian optimization 으로 학습 시간 줄임

RandAugment (RA, 2020)

- 랜덤성에 기반해 매번 다른 augmentation을 사용한다는 철학
- 코드로 구현하기 쉬움

05 참고자료

논문

- A survey on Image Data Augmentation for Deep Learning (2019)
- Improved Regularization of Convolutional Neural Networks with Cutout (2017)
- Mixup: Beyond Empirical Risk Minimization (Image Classification) (2018)
- CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features (Image Classification) (2019)
- AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty (2019)
- Puzzle Mix: Exploiting Saliency and Local Statistics for Optimal Mixup (2020)

유튜브

- 김용환-Asurvey on Image Data Augmentation for Deep Learning
  - https://www.youtube.com/watch?v=TioeCk3yMCo
- [Open DMQA Seminar] Optimal Augmentation
  - https://www.youtube.com/watch?v=5MgxPsEMGY4
- [Open DMQA Seminar] Image Augmentation - Adversarial Learning-Based Methods
  - https://www.youtube.com/watch?v=ItBSS3_zFQU&list=PLpIPLT0Pf7IoTAvBJ6FX1vAi-PIeSw9xK&index=3&t=1905s
- [Open DMQA Seminar] Mixed Sample Data Augmentation
  - https://www.youtube.com/watch?v=AWyiLk3nQ28&list=PLpIPLT0Pf7IoTAvBJ6FX1vAi-PIeSw9xK&index=70
- Tackling the Small Object Problem in Object Detection
  - https://www.youtube.com/watch?v=WeQcURbHA7U
- [2021] A Normalized Gaussian Wasserstein Distance for Tiny Object Detection paper explained
  - https://www.youtube.com/watch?v=eGKlg4sZ0Zw

05 사용기술

라이브러리 및 프레임워크

roboflow

OpenCV

PyTorch

gradio

PyTorch Lightning

Transformers

Albumentations

개발환경

GitHub

분석언어

YAML

python

{JSON}

Q & A

감사합니다

Data Universe

박주경 https://github.com/likespike

김성호 https://github.com/RTYYY

김용훈 https://github.com/dydgns94

윤예은 https://github.com/yenny2

이승주 https://github.com/llukaLee