병변 검출 AI 경진대회

KUBIG CV Team 1: 문성빈 임정준 천원준 황민아

Table of contents

01

FsDet

Few-shot fine-tuning & Training

02

Yolov5

Yolov5-S/L Training

03

Ensemble

Weighted Boxes Fusion

04

Result

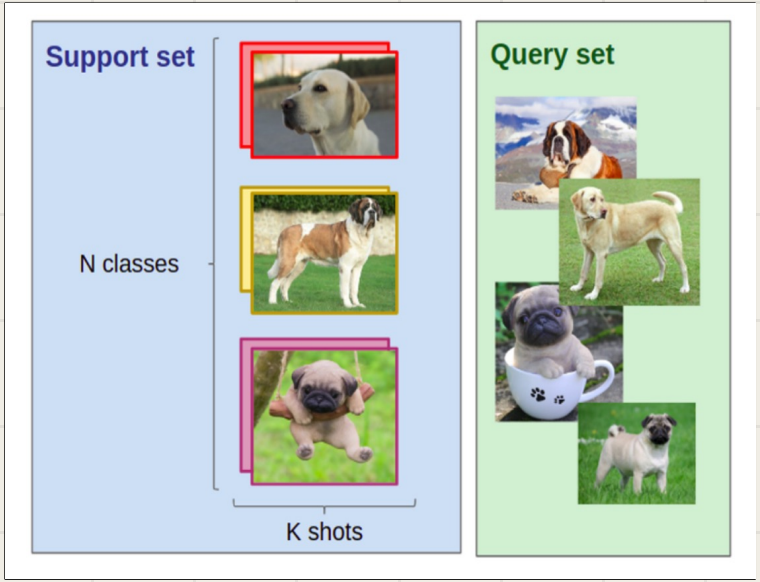
Dacon LeaderBoard & Result

01

Few-Shot Object Detection (FsDet)

Few-Shot Object Detection for Our Dataset

Few-Shot Learning

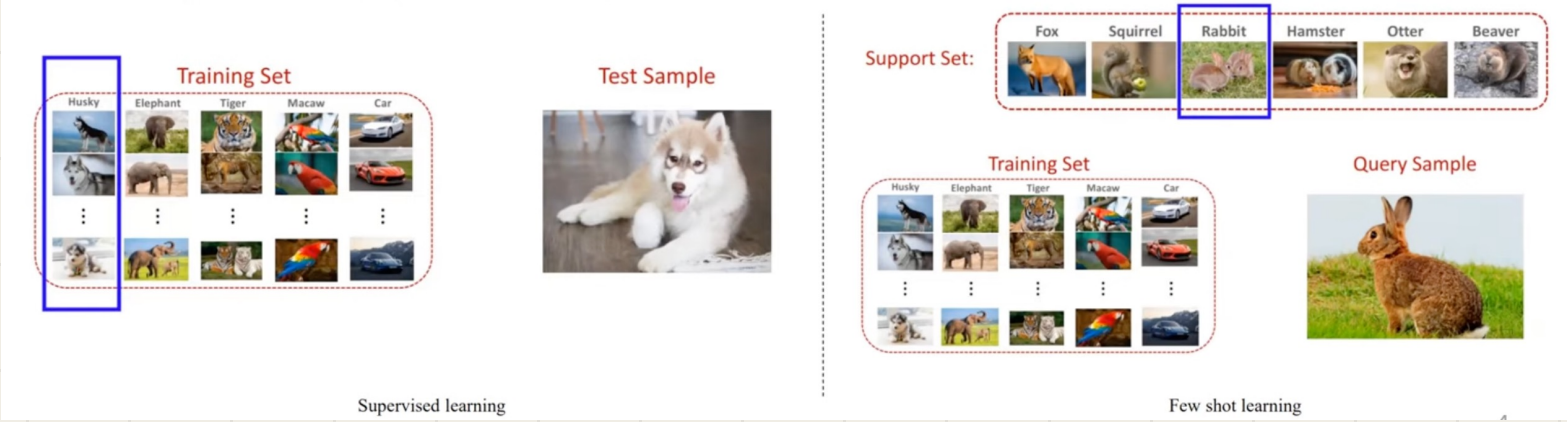


Few-Shot Learning : 매우 **적은 samples**들을 기반으로 classification 또는 regression을 진행

- Supervised Learning은 labeled data를 많이 갖고 있어야 좋은 성능을 낸다 -> **많은 양의 시간과 costs.**
- DL models는 various objects에 대한 **일반화**를 잘하도록 학습시켜야 한다.

-> 이에 **Few-Shot Learning**은 ***Data Collection, Labeling Cost, Generalization Ability***의 어려움에서 이점을 가지는 Learning Methods이다.

Few-Shot Learning



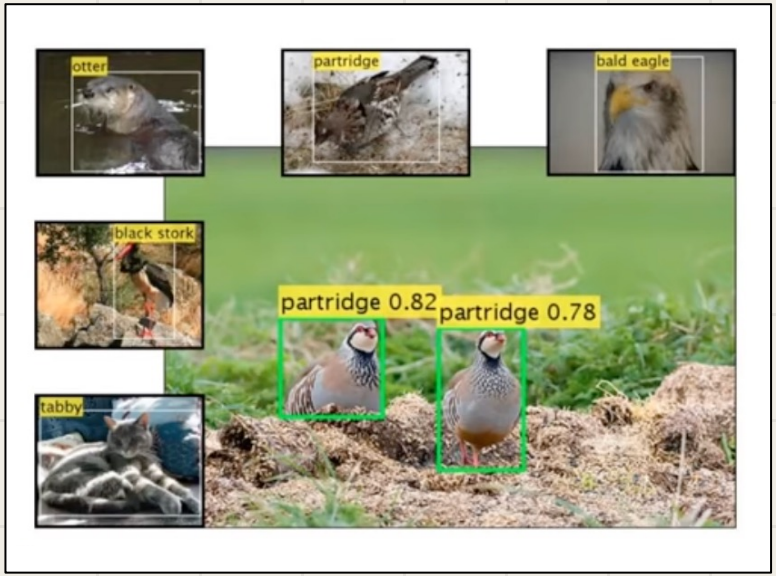
Data – ***Training Set, Support Set, Query Set***

- **Training Set** : For training base classes
- **Support Set** : For training novel classes
- **Query Set** : For testing novel classes

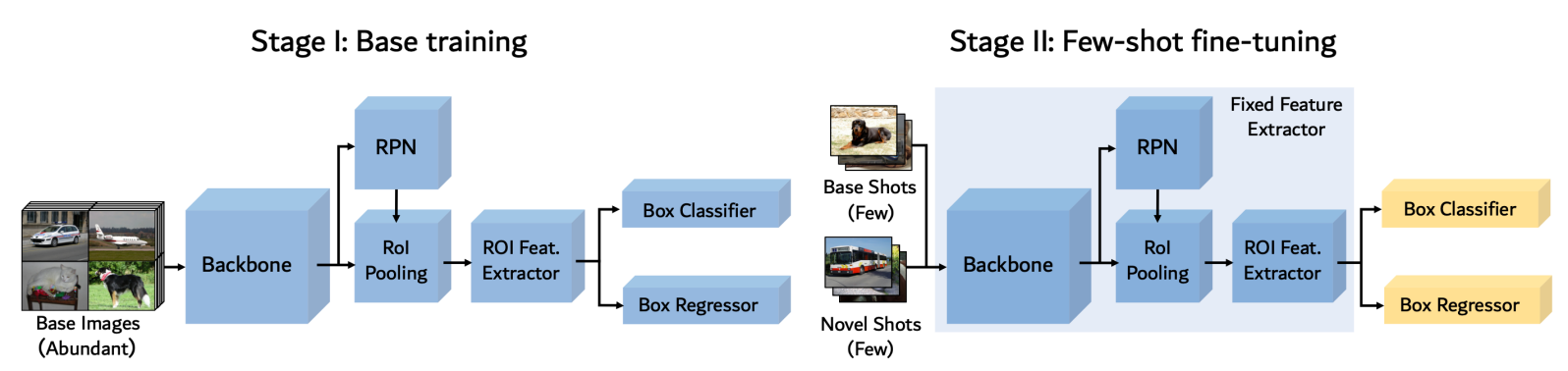
Learning Methods

- **Model-based Approach** : Similarity calculation between feature vectors, regularization to the small data
- Data-driven Approach : Transformation of the Support Set, data generation by GAN

- **Transfer Learning** : Fine-Tuning
- Meta Learning : Learn to Learn



FsDet - Two Stage Fine Tuning Approach



- **Stage I** : Training Object Detector(Fast R-CNN etc..)

- **Stage II** : Detector의 마지막 layer를 적은 양의 base, novel class data를 통해 fine-tune을 진행합니다. 나머지 parameters는 freeze합니다.

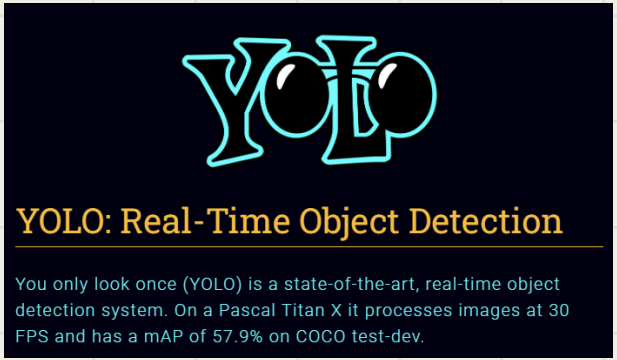
02

YOLOv5

YOLOv5-S/L Training

YOLO

- You Only Look Once
- Darknet을 기반으로 하는 Detection model
- One-stage detection 기법 사용,
  실시간 객체 탐지가 가능



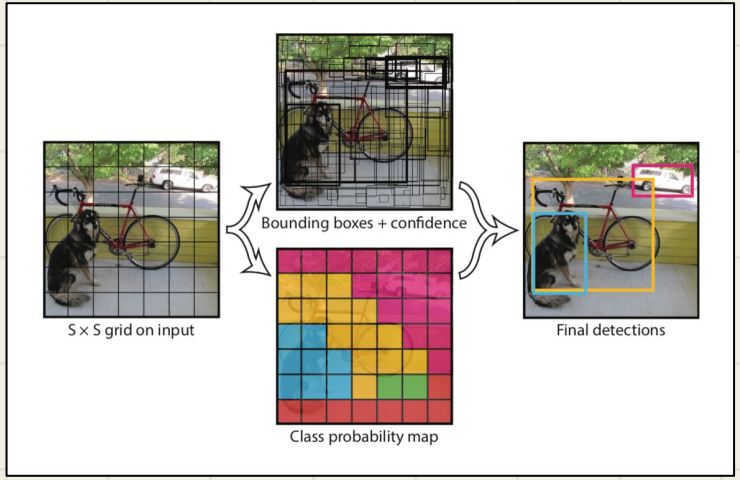
One-Stage Detector

- 1 stage pipeline
- YOLO, RetinaNet, ...

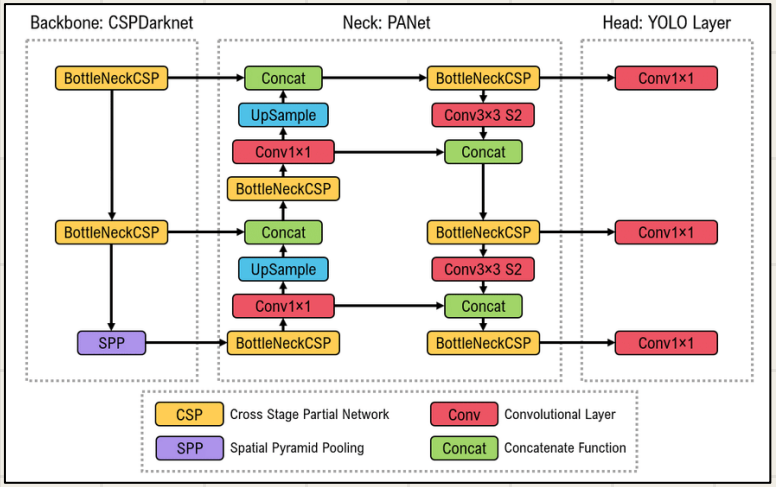
Multi-Stage Detector

- 2 stage pipeline
- RCNN, Faster RCNN, ...

YOLO



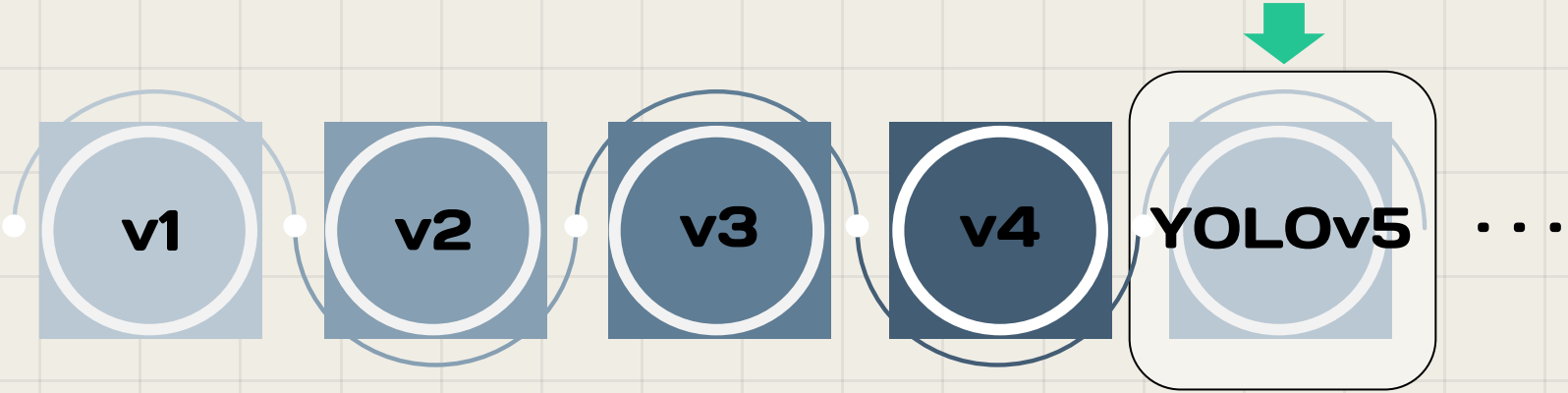
YOLOv5 architecture



- Input 이미지를 S x S grid로 분할
- 각 grid cell마다 bounding box와 confidence score 도출

- Backbone: 이미지에서 feature map 추출
- Head: feature map 바탕으로 물체 위치 찾음

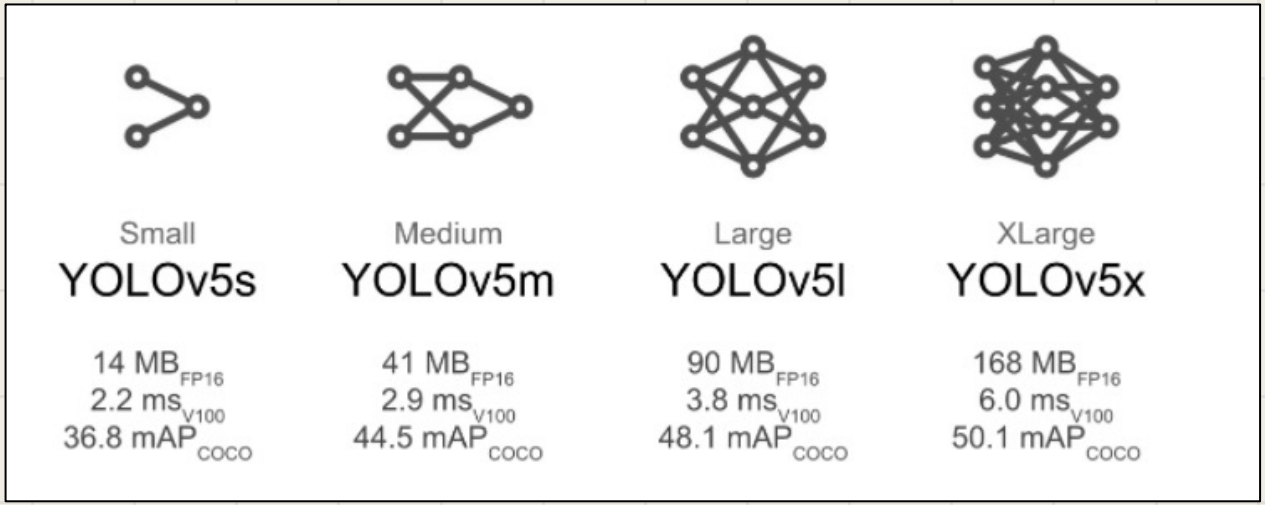
YOLOv5



- Pytorch 기반
- Backbone으로 CSP-Darknet을 사용
- Depth multiple과 width multiple을 기준으로 Backbone 분류

https://github.com/ultralytics/yolov5 를 clone하여 사용

YOLOv5 - small/large



YOLOv5s

Colab GPU

YOLOv5l

Kaggle GPU

Depth multiple, Width multiple에 따른 분류

- Depth multiple이 클수록 BottleneckCSP 레이어가 더 많이 반복되어 더 깊은 모델이 되며
- Width multiple이 클수록 해당 레이어의 conv 필터 수가 많아진다.

Training

- Train/Test를 8:2로 분리하여 Validation 진행
- Hyperparameter:
  - learning_rate=0.01,
  - momentum=0.937(Optimizer='SGD'),
  - weight_decay=0.0005,
  - iou_threshold=0.2,
  - batch_size=64,
  - epoch=30
- 원래 이미지 사이즈는 576x576이나, 모델 Augmentation 및 GPU 고려 Resize 후 진행

03

Ensemble

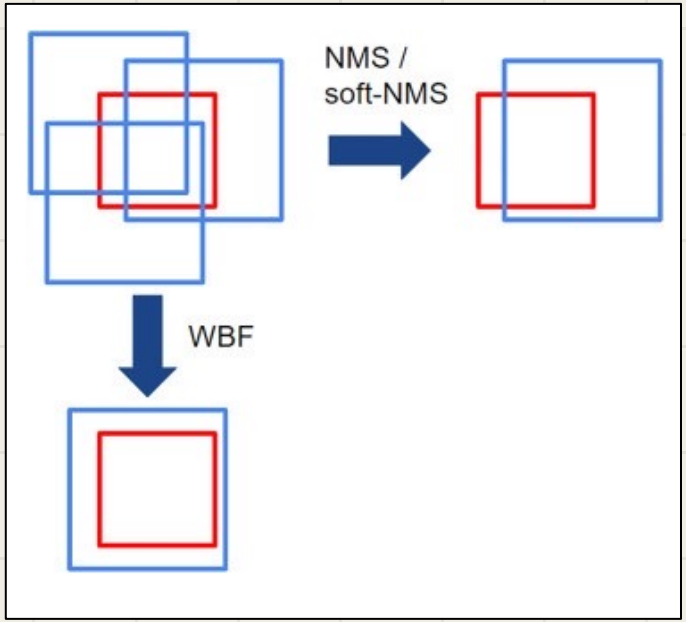
Weighted Boxes Fusion

Weighted Boxes Fusion

- 보통의 앙상블에서는 NMS/Soft-NMS Extension을 통해 예측의 일부를 제거하고 Bounding Box 생성

- WBF(Weighted Boxes Fusion)은 각 모델에서 예측된 Bounding Box 정보를 모두 활용하여 문제를 해결

- 부정확한 박스들이 예측될 때, 이에 대한 가중평균으로 Bounding Box를 생성하여 성능 향상

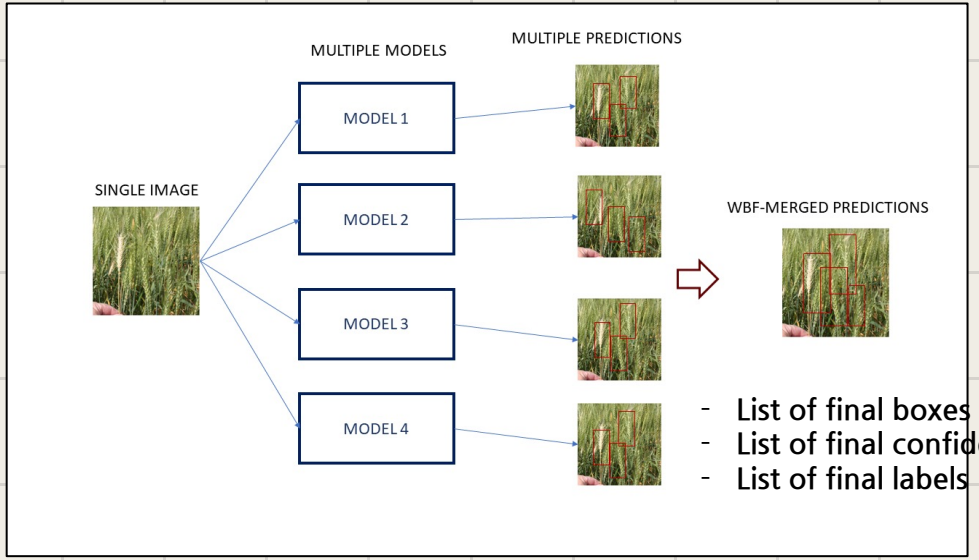


WBF Input & Output

Input

- List of predicted boxes of each model
- List of scores of each model
- List of labels of each model
- List of weights of each model
  (특정 모델에 가중치 더 주기 가능)
- IoU threshold(To check overlap)
- Skip threshold(To discard inaccurate boxes)

Output



- List of final boxes
- List of final confidence scores
- List of final labels

WBF Algorithm

$$\mathbf{C} = \frac{\mathbf{C}_1 + \mathbf{C}_2 + ... + \mathbf{C}_T}{T}, \quad (1)$$

$$\mathbf{X1} = \frac{\mathbf{C}_1 * \mathbf{B_X1}_1 + \mathbf{C}_2 * \mathbf{B_X1}_2 + ... + \mathbf{C}_T * \mathbf{B_X1}_T}{\mathbf{C}_1 + \mathbf{C}_2 + ... + \mathbf{C}_T} \quad (2)$$

$$\mathbf{X2} = \frac{\mathbf{C}_1 * \mathbf{B_X2}_1 + \mathbf{C}_2 * \mathbf{B_X2}_2 + ... + \mathbf{C}_T * \mathbf{B_X2}_T}{\mathbf{C}_1 + \mathbf{C}_2 + ... + \mathbf{C}_T} \quad (3)$$

$$\mathbf{Y1} = \frac{\mathbf{C}_1 * \mathbf{B_Y1}_1 + \mathbf{C}_2 * \mathbf{B_Y1}_2 + ... + \mathbf{C}_T * \mathbf{B_Y1}_T}{\mathbf{C}_1 + \mathbf{C}_2 + ... + \mathbf{C}_T} \quad (4)$$

$$\mathbf{Y2} = \frac{\mathbf{C}_1 * \mathbf{B_Y2}_1 + \mathbf{C}_2 * \mathbf{B_Y2}_2 + ... + \mathbf{C}_T * \mathbf{B_Y2}_T}{\mathbf{C}_1 + \mathbf{C}_2 + ... + \mathbf{C}_T} \quad (5)$$

Preprocess

- Results After NMS
- Normalization: box ÷ Image size

Prefilter boxes

- Merge the bounding box with same label of different model
- Discard if lower than skip_threshold

IoU Calculation

- Fuse the box if iou>=iou_threshold
- Box * score / fusing score

Frame scoring

- Score Manipulation
- Multiple ‘weight’

So we are.....

- <Ensemble>
  - ❑ FsDet 576
  - ❑ Yolov5-L 384
  - ❑ Yolov5-L 512
  - ❑ Yolov5-S 512
  - ❑ Yolov5-S 576

  [2,2,2,1,1] Weight

- Ensemble_boxes 패키지 활용

- 30000 rows로 제출제한
  IoU_threshold=0.53으로 설정하여 진행

```
for t in box2.index:
  label2.append(box2['class_id'][t])
  score2.append(box2['confidence'][t])
  point2.append([box2['point1_x'][t]/575,box2['point1_y'][t]/575,box2['point3_x'][t]/575,box2['point3_y'][t]/575])

for q in box3.index:
  label3.append(box3['class_id'][q])
  score3.append(box3['confidence'][q])
  point3.append([box3['point1_x'][q]/575,box3['point1_y'][q]/575,box3['point3_x'][q]/575,box3['point3_y'][q]/575])

for d in box4.index:
  label4.append(box4['class_id'][d])
  score4.append(box4['confidence'][d])
  point4.append([box4['point1_x'][d]/575,box4['point1_y'][d]/575,box4['point3_x'][d]/575,box4['point3_y'][d]/575])

box_list=[point0,point1, point2,point3,point4];label_list=[label0,label1, label2,label3,label4];score_list=[score0,score1, score2,score3,score4]
boxes, scores, labels=weighted_boxes_fusion(box_list, score_list, label_list, weights=[2,2,2,1,1], iou_thr=0.53, skip_box_thr=0.52)
boxes=boxes*575
for k in range(len(labels)):
  result['file_name'].append(file_name)
  result['class_id'].append(int(labels[k]))
  result['confidence'].append(round(scores[k],4))
  result['point1_x'].append(int(boxes[k][0]))
  result['point1_y'].append(int(boxes[k][1]))
  result['point2_x'].append(int(boxes[k][2]))
  result['point2_y'].append(int(boxes[k][1]))
  result['point3_x'].append(int(boxes[k][2]))
  result['point3_y'].append(int(boxes[k][3]))
```

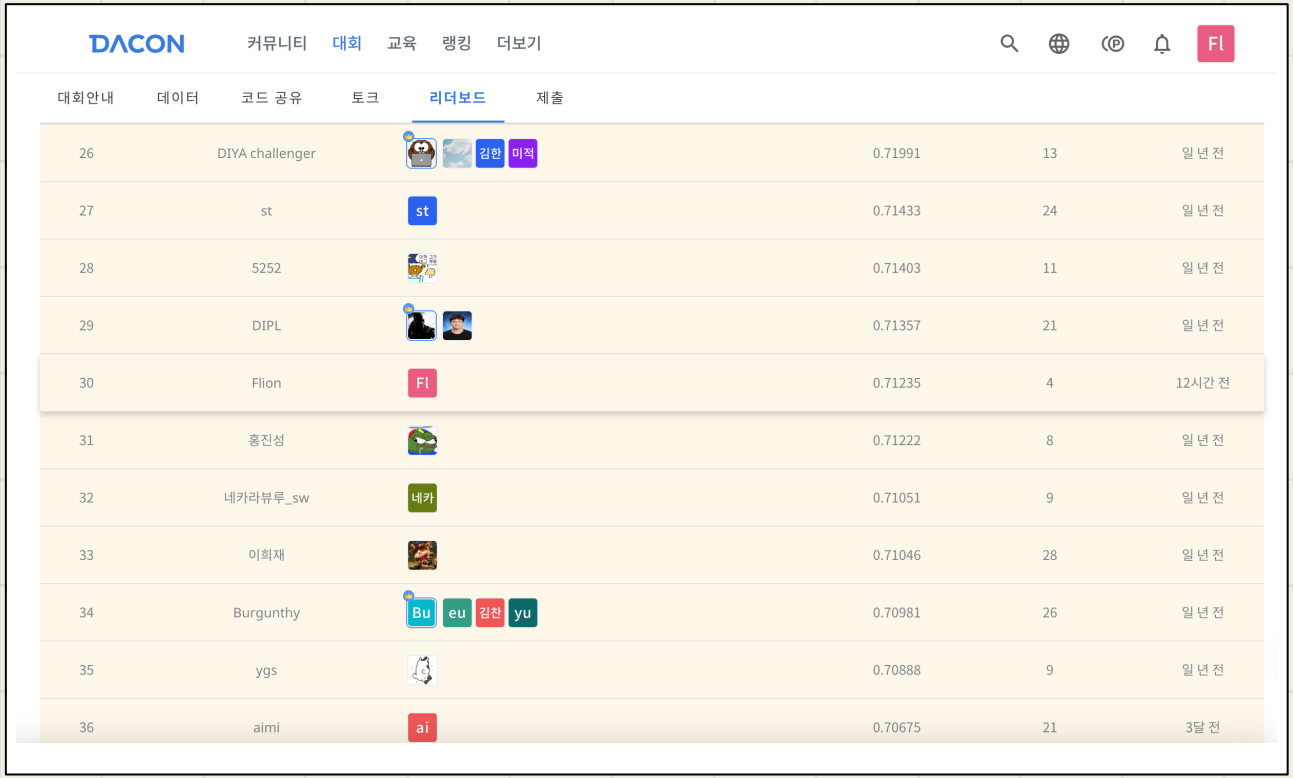
04

Result

Dacon LeaderBoard & Inference

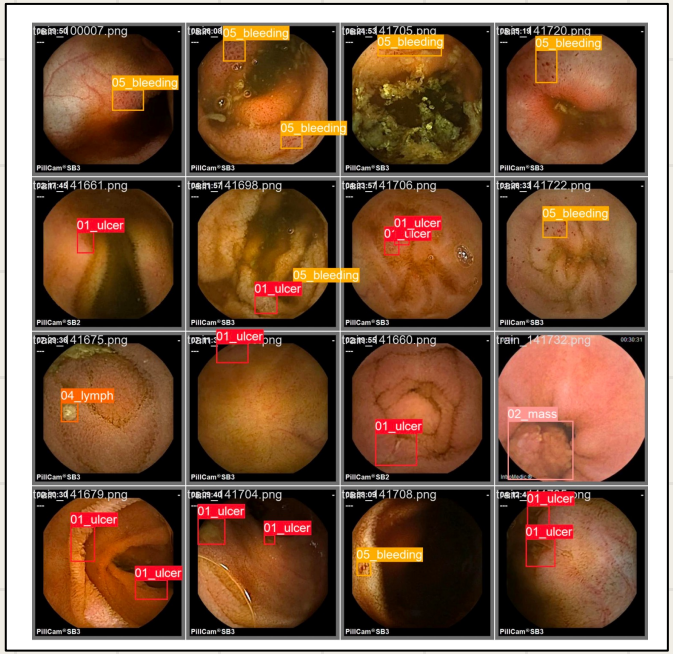
Dacon Leaderboard

- mAP50 기준:
  Public 0.71235
  Private 0.77001
- 361팀 중 30등:
  상위 8%
- 연습대회이긴 하나,
  무료 Colaboratory &
  새로운 모델 활용으로
  괄목할 만한 성과 달성



26	DIYA challenger	김한 미적	0.71991	13	일 년 전
27	st	st	0.71433	24	일 년 전
28	5252		0.71403	11	일 년 전
29	DIPL		0.71357	21	일 년 전
30	Flion	Fl	0.71235	4	12시간 전
31	홍진성		0.71222	8	일 년 전
32	네카라뷰루_sw	네카	0.71051	9	일 년 전
33	이희재		0.71046	28	일 년 전
34	Burgunthy	Bu eu 김찬 yu	0.70981	26	일 년 전
35	ygs		0.70888	9	일 년 전
36	aimi	ai	0.70675	21	3달 전

Inference Result



16기 천원준

- ❑ Baseline(Mask R-CNN)
- ❑ Presentation

16기 임정준

- ❑ Yolov5 - Large Training
- ❑ WBF Ensemble & Submission

17기 문성빈

- ❑ FsDet Training
- ❑ Dataset Debugging

17기 황민아

- ❑ Yolov5 - small Training
- ❑ Preprocessing

Inference result for Validation set of Yolov5 – Large model. **For more info, please refer to KUBIG Github**

Thanks!

Do you have any questions?

kubigkorea@gmail.com
github.com/ku-big
notion c11.kr/kubig