

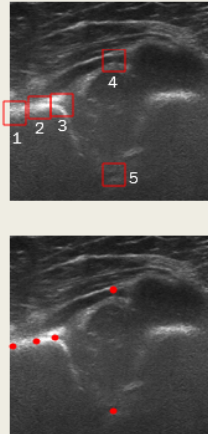


이서연 수행 task 최종 정리 _240425

0. 연구자 : 이서연 (영문 이름 : Lee seoyeon) , 고려대학교 바이오의공학부 학부 재학

1. 목표 : 2 stage detection

- Global detection
 - 5개 영역의 검출을 위한 사전 검출 AI
- Local detection
 - Global detection에서 예측된 위치에서 정확한 지점을 찾는 AI



[과제 수행 내용 영문 정리]

The goal is to detect tumor's location by conducting a 2 stage detection.

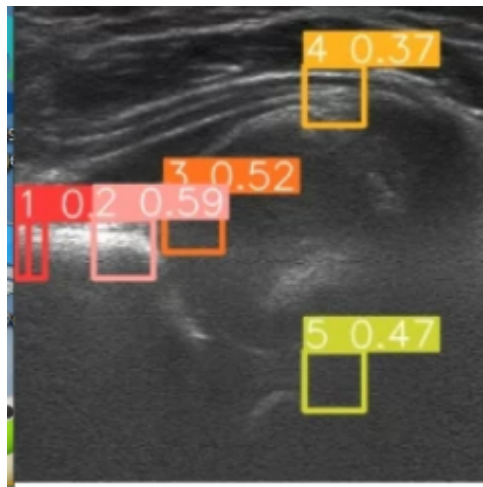
We utilize the YOLOv8 architecture with inputs sized 256×256 to detect 5 tumor regions. For each input, we predict the label and region of each point.

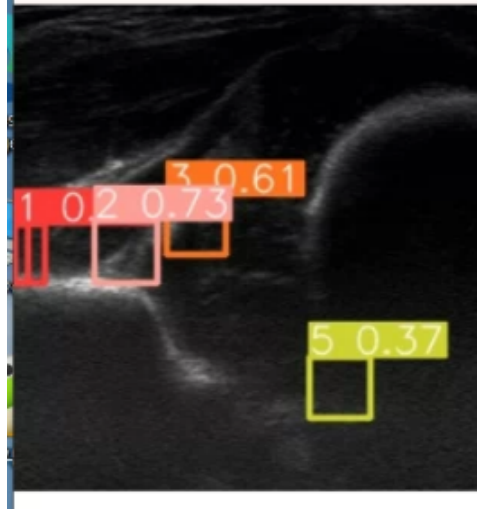
Subsequently, We conduct a local detection to localize the exact points of the tumor based on the results of global detection. In this process, we utilize a Multi-Layer Perceptron(MLP) to predict the two points ; points on tumor's location.

The mean IoU of five points obtained from a global detection is 0.85, indicating improved results compared to the previous model.

1. global detection :

- 수행 task 정리
 - yolo labels : width, height 32
 - yolov8x 모델 사용 , 주어진 데이터로 weigth 학습 , epochs=150 batch=16 imgsz=640
 - yolov8x_class5.weight 사용해 predict 진행 (predict)





◦ IoU

- 테스트 이미지의 모든 predicted point 대해서 GT와의 IOU값을 구해서 csv로 저장(ious5.csv 생성)

	Image_Name	label	IoU
0	1000260.txt	2	0.990026
1	1000260.txt	3	0.987980
2	1000260.txt	1	0.243137
3	1000260.txt	5	0.984715
4	1000260.txt	4	0.980112
5	1000260.txt	1	0.500819
6	1001693.txt	2	0.984206
7	1001693.txt	4	0.986367
8	1001693.txt	3	0.991786
9	1001517.txt	2	0.996289
10	1001517.txt	4	0.983831
11	1001517.txt	1	0.243186
12	1001517.txt	3	0.991697
13	1000492.txt	2	0.984764
14	1000492.txt	3	0.984016
15	1000492.txt	5	0.984558
16	1000492.txt	4	0.989075
17	1000492.txt	1	0.244321
18	1000492.txt	1	0.493362
19	1000492.txt	4	0.798627

- 결과 : point 1의 경우 다소 low한 결과(구석진 곳에 있어서 그런 것으로 예상) , mean IoU의 경우 0.85로 기존의 모델에 비해 향상된 결과를 얻어냄

	Label	Mean_IoU
0	1	0.358205
1	2	0.974353
2	3	0.964319
3	4	0.947752
4	5	0.982439

mean IoU of 5 points: 0.845413683706712

추가 자료

→ Global Detection의 IoU

영역 별 IoU		1	2	3	4	5
종류						
ResNet50	<u>IoU avg.</u>	0.896	0.818	0.853	0.769	0.724
	<u>IoU std.</u>	0.062	0.088	0.074	0.149	0.142
EfficientNetB2	<u>IoU avg.</u>	0.941	0.900	0.916	0.838	0.832
	<u>IoU std.</u>	0.039	0.061	0.046	0.106	0.095
Custom Model	<u>IoU avg.</u>	0.643	0.542	0.502	0.419	0.432
	<u>IoU std.</u>	0.194	0.183	0.183	0.209	0.190

2. local detection :

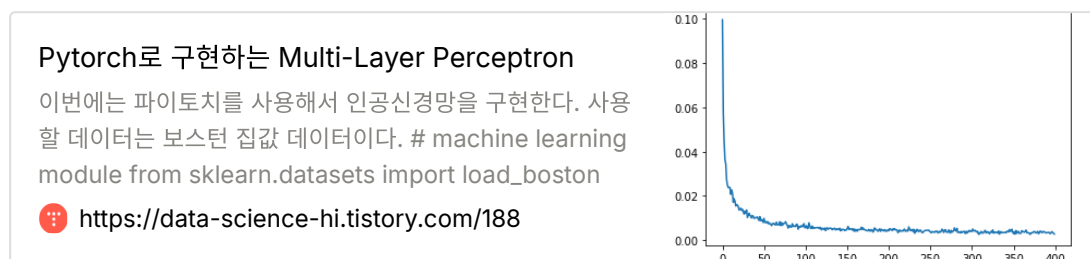
- 수행 task 정리 :
 - test data set을 local detection의 train test로 사용함.

	image_name	label	x_predict	y_predict	width	height	x_real	y_real
0	1000260	2	0.230820	0.507169	0.125156	0.125906	0.230469	0.507812
1	1000260	3	0.374602	0.453481	0.124248	0.124245	0.375000	0.453125
2	1000260	1	0.015297	0.512720	0.030526	0.125199	0.000000	0.511719
3	1000260	5	0.668618	0.786438	0.124875	0.125470	0.667969	0.785156
4	1000260	4	0.668844	0.190538	0.125106	0.126652	0.667969	0.191406
5	1000260	1	0.031403	0.512270	0.062805	0.125041	0.000000	0.511719
6	1001693	2	0.230622	0.507320	0.125628	0.126371	0.230469	0.507812
7	1001693	4	0.667871	0.191968	0.123847	0.125238	0.667969	0.191406
8	1001693	3	0.374996	0.452756	0.124340	0.124719	0.375000	0.453125
9	1001517	2	0.230661	0.507676	0.125012	0.125273	0.230469	0.507812

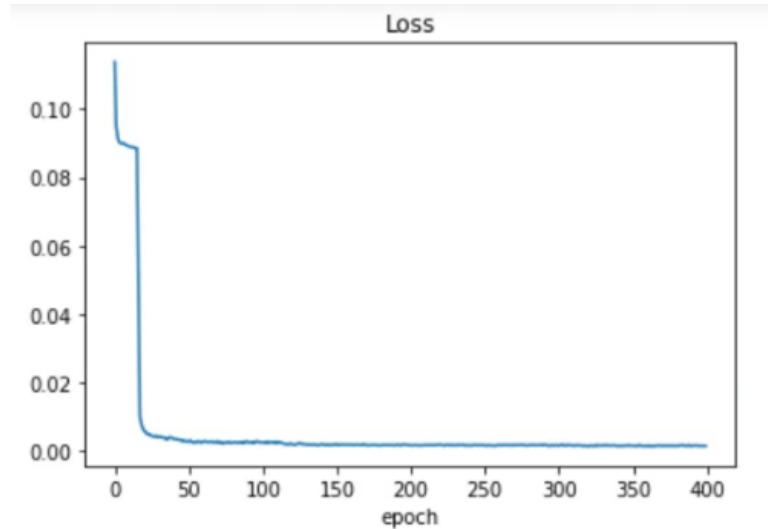
3031

→ 이미지별로 5개 point를 완벽하게 예측 못한 경우도 있어서 x_predict , y_predict 와 실제 이미지의 5개의 point의 거리를 구해 가장 가까운 값을 x_real, y_real로 생각하고 data 생성

- MLP 사용 : 위의 df를 train, test split (8:2)로 나누어 진행 , epoch=400



- col_x(input) 으로 col_y(output)을 예측
 - col_id = ['image_name']
 - col_x = ['x_predict', 'y_predict', 'width', 'height']
 - col_y = ['x_real', 'y_real']
- loss



- step 2의 test data 를 가지고 최종 산출물 생성 rmse 계산 _
(predict_point1.csv생성)

	img	label	predict_x	predict_y	real_x	real_y	rmse
1948	1000911	1	0.000000	0.508959	0.000000	0.511719	0.002760
1949	1000911	4	0.655044	0.208197	0.667969	0.191406	0.021189
1950	1000911	1	0.000000	0.510090	0.000000	0.511719	0.001629
1951	1000911	2	0.229085	0.502264	0.230469	0.507812	0.005718
1952	1001293	2	0.245295	0.502155	0.230469	0.507812	0.015869
1953	1001293	3	0.376668	0.464241	0.375000	0.453125	0.011241
1954	1001293	1	0.000000	0.510040	0.000000	0.511719	0.001678
1955	1000687	1	0.000000	0.508938	0.000000	0.511719	0.002780
1956	1000687	3	0.377424	0.463515	0.375000	0.453125	0.010669
1957	1000687	2	0.245019	0.502157	0.230469	0.507812	0.015611

- point 별로 mean_rmse 및 최종 rmse 출력

```

Label  Mean_mse
0      1  0.002194
1      2  0.015273
2      3  0.012466
3      4  0.023721
4      5  0.003925
mean RMSE of 5 points: 0.011515494874084478

```

- 예측하는 데 있어 예측 값과 유사한 x_center, y_center을 집어 넣었기 때문에 정확도가 높은 center 값이 나왔다고 여겨짐. 발표자료를 참고하면 크롭 이미지를 넣어 학습하는 것이었는데 그렇게 하지는 않았음.
- 실제로 하나의 이미지에 대해 예측값과 GT 값을 비교해봄

