Data-Centric CV WrapUP Report

Haneol Kim, Bohyun Kim, Sungjoo Kim, Suhyun Jung, Namgyu Yun, Minseok Heo [1]

[1] *Naver Boostcamp AI Tech, CV 21조*

1 프로젝트 개요

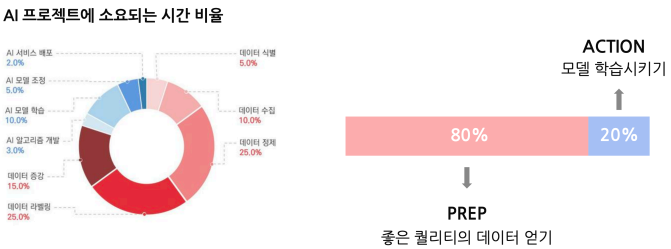

Figure 1: Data-Centric의 등장 배경

Data-Centric 접근을 기반으로 데이터 품질을 개선하고, 글자 검출 및 인식 성능을 강화하는 것을 목표로 했다. 품질 높은 데이터를 쉽게 얻기는 어렵고, 모델 성능에 직접적인 영향을 미치기 때문에 신중하게 준비해야 한다.

이로 인해 최근 AI 시스템의 성공을 위해 'Data-Centric AI' 접근이 필요하다는 목소리가 높아지고 있다. 데이터 문제는 단순히 모델의 성능에만 영향을 미치는 것이 아니라, AI 시스템의 신뢰성과 효율성, 그리고 비즈니스의 결과 전반에 걸쳐 영향을 미친다. 따라서 성공적인 AI 시스템을 위해서는 모델 성능 개선에만 집중하는 것이 아니라, 높은 품질의 데이터 확보와 검증을 통한 Data-centric 접근이 반드시 필요하다. 프로젝트를 통해 모델링을 통한 성능 향상이 아닌 데이터를 통한 성능 향상을 하고자한다.

2 프로젝트 팀 구성 및 역할

이름	공통역할	개인역할
김한얼	EDA	Construct pipeline, Code refactoring, Schedule management, Workload management
김보현		Relabeling by CVAT, Construct pipeline, Data augmentation
김성주		Data feature analysis, Enhanced evaluation accuracy, Implemention data augmentation technique
윤남규		Research tools, Hypothesis test, Data curation & augmentation, Pipeline refatorization
정수현		Synthetic data, Hypothesis test, Data augmentation
허민석		Hypothesis test, Hyperparameter tuning, Data augmentation

Figure 2: 역할

공통 역할은 annotation guideline을 세워서 EDA를 했다. EDA가 끝난 후, 개인 역할을 분담하여 프로젝트를 진행했다.

3 프로젝트 수행 절차 및 방법

3.1 계획 설정

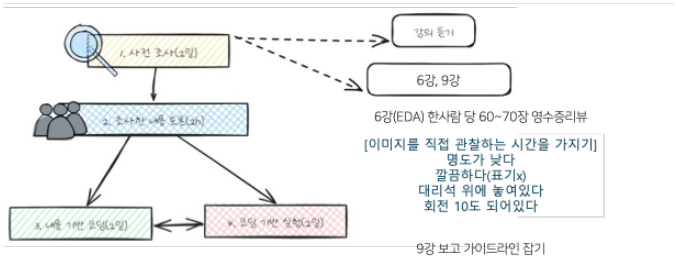

Figure 3: 초기 계획 설정

지난 프로젝트를 거치며 초기에 체계를 설정하는 것의 중요성을 깨달았다. 시작할 때, workflow와 deadline을 잡아서 체계적으로 프로젝트를 진행하자는 목표를 세웠다. 전체적으로는 가설세우기-토론-코딩-실험 과정으로 진행하겠다는 계획을 세웠고 세부 계획은 그날 데일리 스크럼에서 to do list를 개인별로 부여해서 피어세션 시간에 보고 및 피드백을 받기로 했다.

3.2 EDA

프로젝트가 시작되고 첫째 날과 그 다음날까지 강의를 모두 들었다. 강의에서 들은 내용을 바탕으로 annotation guideline을 세워서 인당 80장씩 EDA를 했다.

Figure 4: EDA

스캔본임, 확대해도 흐릿함, 박스가 잘못됨, 그림자가 존재함, 손가락이 영수증을 침범함, QR코드나 바코드가 존재함, 박스가 겹침, 텍스트가 로고나 QR코드 안에 있음, 글씨가 잘림, 글씨가 휘어짐, 세로 글씨가 존재함, 기울어짐으로 총 9가지의 태그를 만들었다.

3.3 목표 설정

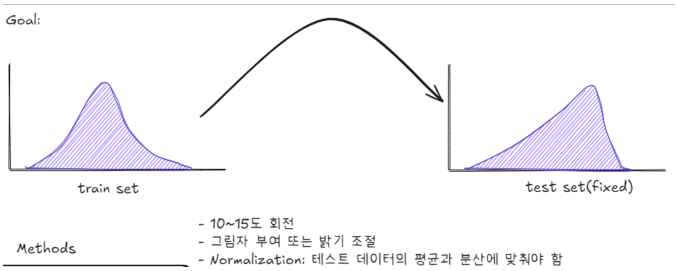

Figure 5: normalize

test data는 수정해선 안되기 때문에, train의 분포를 test에 맞게 normalize해야한다고 생각했다. test data를 계산한 결과,

Mean:(0.683, 0.657, 0.624)

Standard Deviation:(0.198, 0.205, 0.211)

평균과 분산값을 얻을 수 있었고 이를 dataset.py에 반영했다.

3.4 format 변환

각 언어별로 100장씩 있던 train데이터를 8:2의 비율로 train과 validation 데이터로 분리하고 기본 baseline 코드를 실행하였을 때 모델이 학습하는 데 걸리는 시간이 8시간 정도 소요가 되었다. 다양한 실험을 빠르게 진행하기 위해서 train의 이미지 파일들을 pickle 형태로 바꾸었다. pickle형태로 바꾼 언어 별로 train데이터 80개를 바탕으로 학습을 진행하니 4시간으로 2배 정도 시간이 줄게 되었다.

3.5 실험과정

다양한 augmentation 방법론을 바탕으로 실험을 진행하였다. 3.3에서 얻은 normalize에 대한 평균과 분산 값을 바탕으로 Albumentation 라이브러리의 ColorJitter와 함께 사용하였더니 f1_score가 상승하였다. 이후 이미지를 padding하거나 작은 이미지를 확대해보는 augmentation을 추가하였는데 f1_score가 엄청 많이 하락하였다. 이를 확인해보고자 시각화 하는 작업을 거쳤다. 왼쪽 사진

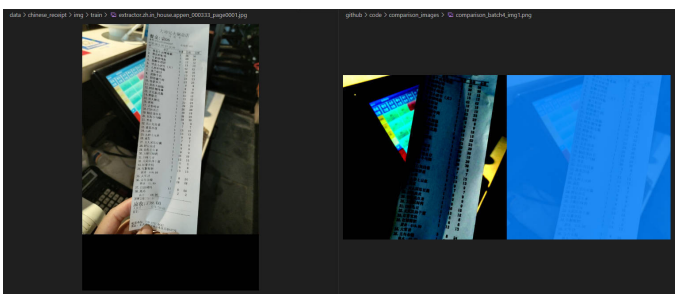

Figure 6: 실험과정

은 원본 사진이고 중간 사진은 pickle로만 바꾼 상태, 제일 오른쪽 사진은 train하기 직전의 사진이다. 그렇다. pickle로 바꾸는 과정에서 augmentaiton이 한 번 일어나고 train하기 전에 또 augmentation이 일어났다는 것을 알게 되었다. 이미지를 pickle로 바꾸는 단계에서만 augmentation을 진행하기로 하였고 여러 augmentation을 진행해보는 실험을 가졌다. (image_size조절, Color Jitter, Normalize, Blur, RandomShadow, RandomBrightnessContrast, binarization 등) 사진은 왼쪽부터 GaussianNoise, Blur, Normalize,

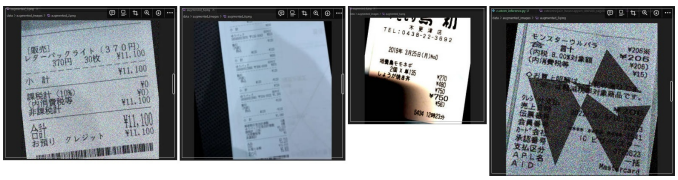

Figure 7: ensemble

RandomShadow→GaussianNoise 이다.

3.6 방법론

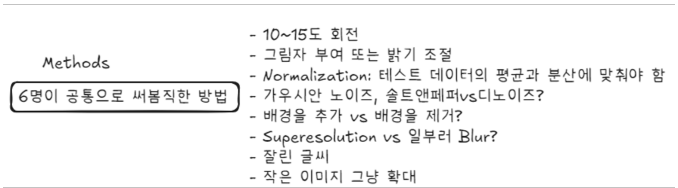

Figure 8: method

train data가 400장밖에 되지 않기 때문에 최대한 pumping해야한다고 생각했다. 따라서 많은 증강 기법들을 적용해보겠다는 계획을 세웠다. data에서 rotate90같은 극단적인 회전은 없었기 때문에 회전 각도는 10도 내외로 하기로 했다.

또한 영수증을 찍는 과정에서 텍스트가 그림자에 가려진 경우가 많았기 때문에 그림자를 추가하거나 밝기 조절해서 없애는 증강 기법도 적용하기로 했다.

종횡비가 성능에 영향을 끼칠 수 있다는 강의 내용을 바탕으로, 패딩을 하거나 그 반대의 상황을 적용해서 성능을 비교해보고자 했다. 가우시안노이즈, 솔트앤페퍼,디노이즈,superreolution,blur 등의 증강도 이후에 추가해보자는 계획을 세웠다.

잘린글씨는 train data에만 있고 test data엔 없기 때문에 train data의 잘린 글씨들을 masking처리하기로 했다.

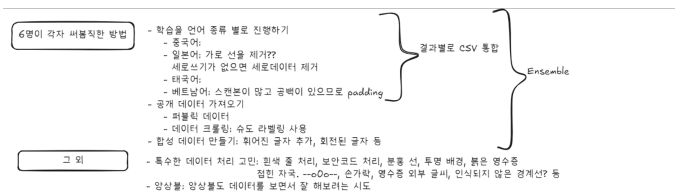

Figure 9: method

언어 별로 학습을 한 후 결과 별로 csv통합을 진행하기로 했다. 그리고 여유가 된다면 한자권의 중국과 일본을 함께 학습해보기

로 했다.

4 Data

4.1 Re-labeling

처음엔 CVAT 공식 문서를 보고 docker에서 cvat를 git clone해와서 작업했지만 super계정을 만들기 까진 성공했는데, 어쩐일인지 local host가 열리지 않았다. docker를 쓰지 않고 웹 기반에서 작업하니 훨씬 수월했다. 배경에 bbox가 생기는 문제, QR 코드에 bbox가 생기는 문제, bbox가 text보다 크거나 작게 그려지는 문제를 해결하는데 중점을 두고 relabeling을 진행했다. 라벨이

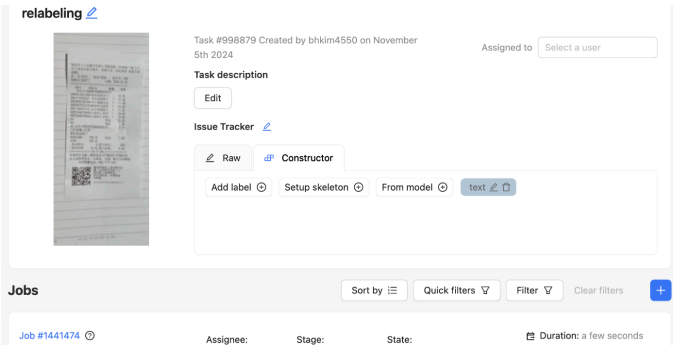

Figure 10: cvat.org 작업화면

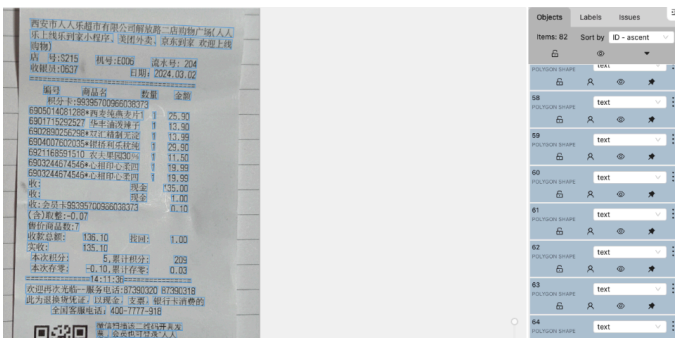

Figure 11: 한 프레임 작업 완료 후

하나만 필요해서(text) 하나만 만들었다. label을 만들고 continue를 눌러준다.그리고 submit & open을 눌러준다. 그러면 그러면 task가 생성되고 몇분 후 위와 같은 화면이 뜨는데 아래쪽의 파란 글씨 여기서는'Job #1441474' 를 누르면 작업을할 수 있는 환경으로 이동한다.

Draw new polygon을 눌러서 원하는 points 개수를 설정해준다. 여기서 주의할 점은 track이 아닌 shape를 눌러줘서 이 프레임에서만 작업해줘야 한다는 점이다. UFO 형식으로 나중에 변환해주려면 4points로 설정해줘야한다. 총 400프레임을 작업하려고 했는데 시간 관계 상 relabeling을 마무리 하지 못하고 합성데이터로 넘어가게 되었다.

4.2 Synthetic Data

아래 figure 처럼 3세트 30장 언어당 총 90장에 합성 데이터를 생성했다. 언어 별로 말뭉치 전처리 파일(.txt 형식), 합성 이미지 생성을 위한 폰트를 통해서 합성 이미지를 만들었다. 만든 합성데이터로 아래 3번의 실험을 진행하려는 계획을 세웠다.

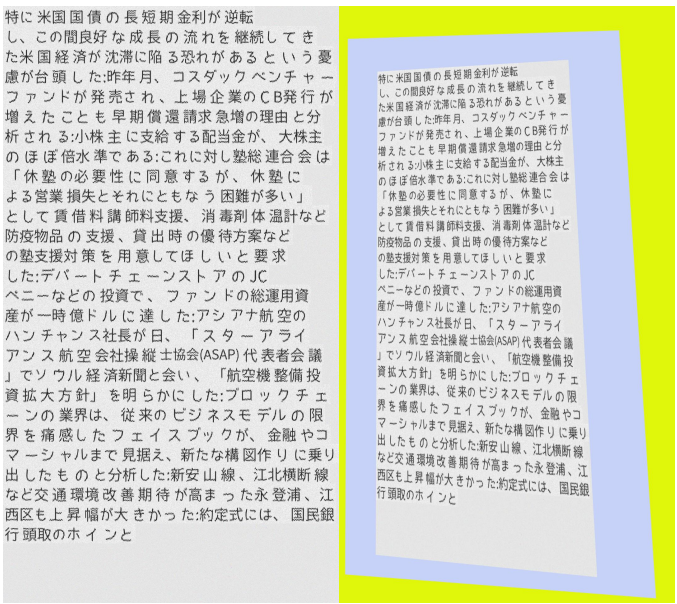

Figure 12: synthetic data

왼쪽은 합성데이터이고 오른쪽은 합성데이터에 증강을 추가한 모습이다. 합성데이터가 주어졌을 때, 기존 주어진 data로만 학습할 때와는 다르게 합성 데이터로 한 번 더 pretraining을 해주어야 한다. 그 다음 기존 data에 대해 fine-tuning을 진행한다.

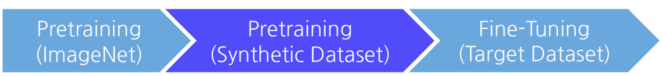

Figure 13: 합성데이터 활용

1. 합성 데이터셋 fine-tuning > 학습된 모델을 ckpt를 불러와 기존 데이터셋 fine-tuning, 2. 기존 데이터셋을 학습한 최고성능 모델 ckpt를 불러와 합성 데이터셋 fine-tuning, 3. 기존+합성 학습

figure 12는 합성한 이미지에 bbox를 표시한 모습이다. 띄어쓰기까지 bbox가 잘 만들어진 모습을 확인할 수 있다.

5 Workflow

실험에 앞서, 코드와 모델에 대한 분석을 먼저 했다. 프로젝트에서는 SceneTextDataset 클래스와 EAST 모델을 사용해서 텍스트 검출을 한다. SceneTextDataset 클래스에서 다양한 증강 기술들을 불러올 수 있었다. 이미지 파일 경로와 필터링을 거친 이미지는 리사이즈, 높이 조정, 이미지 회전, 크롭 등의 전처리 과정을

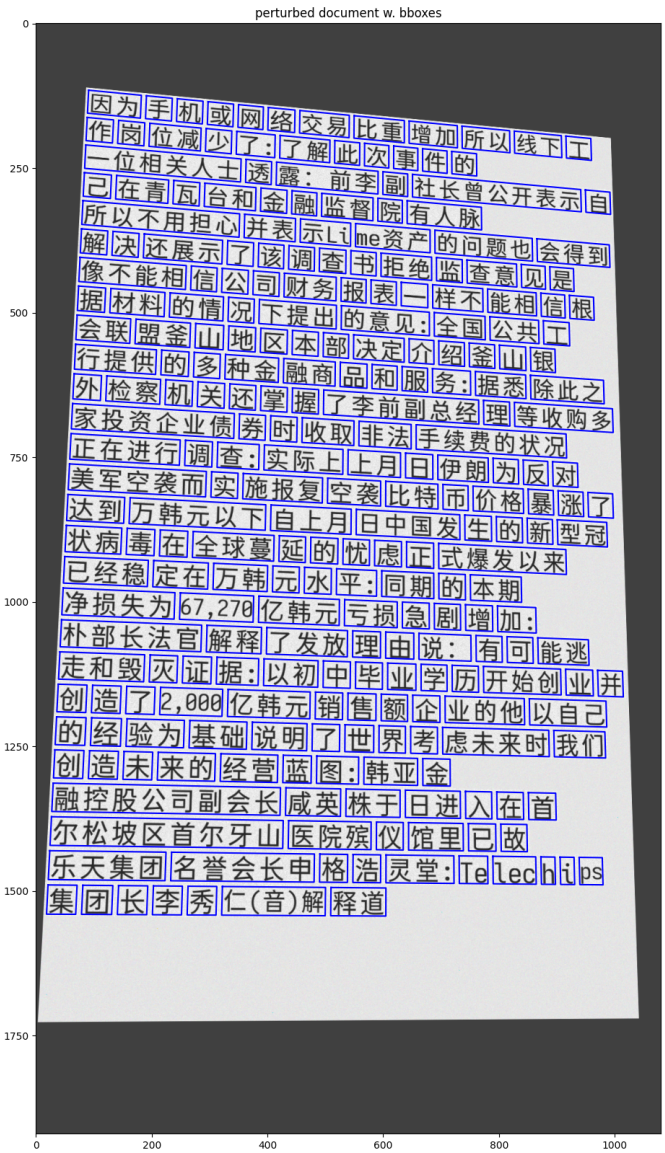

Figure 14: synthetic image with bbox

거친 후 normalize과 jitter를 적용한다. 이제 여기서 RoI mask를 생성해서 학습에 사용할 준비가 완료된다.

그 다음 EAST 모델은 이미지와 스코어맵, 지오맵, 그리고 roi 마스크를 입력으로 받아 텍스트 검출 작업을 수행한다. 모델의 내부에는 Extractor, Merge, Output 모듈이 포함되어 있어 입력 이미지로부터 특징을 추출하고, 예측 스코어맵과 예측 지오맵을 생성하는데 생성된 예측 값들은 EAST 모델의 손실 함수로 전달되어 학습에 필요한 오차를 계산해준다.

train.py 파일의 parse_args 함수를 통해 학습에 필요한 인자를 받아들이고 do_training 함수를 통해 학습이 된다.

json파일을 더 효율적으로 처리하기 위해 pickle format으로 변환하는 코드를 추가했다.

5.1 시행착오

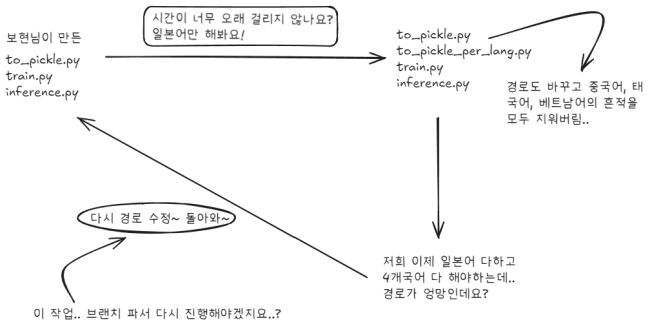

Figure 16: re:제로부터 시작하다

세상에나 이런 일이.. 언어 별로 해보고자 했던 것이 독이 되어 돌아왔다. 포기하긴 아직 이르다! 물론 과거에 잘못 뿌린 씨앗이었지만..내가 거두면 되는법! 결국 마감 전날 밤 새벽에 시간을 투자해서 이뤄내고야 말았다. 결과가 무척 잘 나오는 것은 아니지만 복구 전 f1 score가 0.2대에서 맴돌던 것에 비하면 무척이나 큰 성과라고 볼 수 있다.

6 앙상블

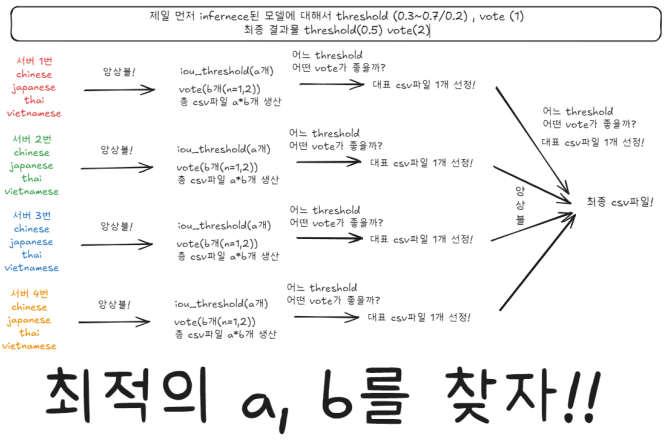

Figure 18: last dance

inference된 모델 결과를 결합하여 최적의 성능을 내기 위해 모인 마지막 날. 특정 IoU threshold 값과 vote 값을 조합하여 다양한 설정의 결과를 만들어 낼 만발의 준비를 마쳤다.threshold 값(0.3 0.7 까지)과 vote 값(1 or 2)을 적용하여 총 a * b개의 결과 csv 파일을 생성하려는 계획을 세웠다. 서버가 각각의 설정을 적용해 예측 결과를 만든 후, 그 중에서 가장 좋은 성능을 보이는 threshold와 vote 조합을 선택해서 제출만 하면 되는 상황이였다.

그러나 ! 우리가 간과한 건 마감시간이였다. 점점 시간은 우리의 목을 조여왔고.. 2시30분부터 진행했던 zoom은 7시까지

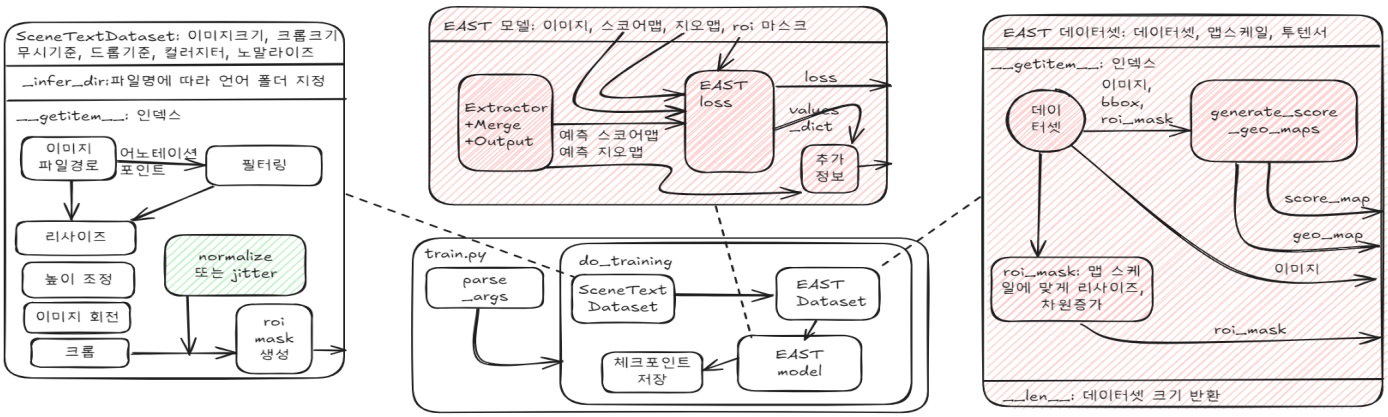

Figure 15: 전 과정 분석도

precision precision (Final)	recall recall (Final)	f1 f1 (Final)	Created at
0.7016 –	0.7978 –	0.7466 –	2024.11.07 01:51

Figure 17: 복구를 완료한 모습

아무도 나가지 못하고 달을 향해 나아가고 있었다. 설령 달에 미치지 못하더라도.. 별들 사이에 있게 되자는 하나의 마음으로.

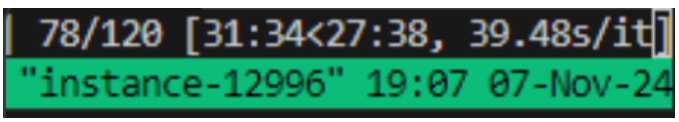

Figure 19: 어라? 어째서 시간이..

내일 지구가 멸망하더라도.. 나는 오늘 한 그루의 사과나무를 심겠다는 마음으로 진행한 앙상블은 사과 열매를 수확해보지 못하고 끝을 맺게 되었다.

그렇게 날개를 펴지 못한 채 꽃봉오리에서 안타깝게 생을 마감하고 말았지만..역풍에 주저하지말고 다음 프로젝트에서 또 다시 날아보도록 하자.. 새 한 마리만 그려넣으면 남은 여백은 모두 하늘인 것을..!

7 Score

증강기법	Only fitter	precision	recall	f1_score
Gaussian Noise(varlimit=(500,5000),p=1.0)	✓	0.81274	0.76856	0.79003
Image_size(1024)	✓	0.79573	0.75539	0.77504
Blur(blur_limit=(7,7),p=0.5)	✓	0.76828	0.75506	0.76161
Jitter(p=0.3) + Normalization	✓	0.77755	0.74321	0.75999
Image_size(2048), crop_size(2048)	✓	0.74446	0.7577	0.75102
Gaussian Noise(varlimit=(500,5000),p=1.0)	✓	0.75486	0.74403	0.74941
B -> GN Blur(p=0.1)	✓	0.75902	0.71177	0.73464
RandomShadow(shadow_roi=(0,0,1,1))	✓	0.74106	0.71967	0.73021
Gaussian Noise(varlimit=(500,5000),p=0.5)	✓	0.72025	0.72807	0.72414
B -> GN	✓	0.73259	0.70123	0.71657
Blur(blur_limit(5,5),p=0.5)	✓	0.70112	0.68938	0.6952
RSH -> GN	✓	0.68	0.68247	0.68123
Normalize+약한jitter	✓	0.6917	0.59878	0.64189
Only Normalization	✓	0.6917	0.59878	0.64189
Image_size(2048)-renormal포함	✓	0.60433	0.57935	0.64189
Jitter(p=1) v.2	✓	0.71396	0.57912	0.63951
GN -> B	✓	0.66251	0.60428	0.63206
Jitter(0.1, p=1) + Normalization	✓	0.68859	0.57935	0.62926
Image_size(1536)-renormal포함	✓	0.63727	0.59878	0.60433
Jitter(p=0.6) + Normalization	✓	0.63465	0.52902	0.57704
Normalize+약한jitter(2)	✓	0.63717	0.51242	0.56588
Jitter(0.1v.2, p=1) + Normalization	✓	0.63179	0.51242	0.56588
Jitter(p=1) + Normalization	✓	0.61452	0.51071	0.55783
Image_size(1024)-renormal포함		0.58451	0.51071	0.55764
Jitter(0.2, p=1) + Normalization	✓	0.6271	0.48484	0.54687
Image_size(1536, 2048)-renormal포함	✓	0.55964	0.48484	0.54687
Jitter(0.15, p=1) + Normalization	✓	0.10332	0.19926	0.13608

Table 1: 앙상블 전 실험 결과들

이번 프로젝트에서의 점수는, 그야말로 "모두의 땀과 눈물의 결정체"라고 할 수 있겠다. 앙상블을 위한 마지막 순간까지 최적의 threshold와 vote 조합을 찾기 위해 전력을 다했지만, 결국 마감시간의 벽을 넘지 못하고 최종 결과 제출까지 닿지는 못했다. 그럼에도 불구하고, 중간에 거친 여러 시행착오와 복구 과정에서 얻은

점진적인 성과는 각자의 성장으로 이어졌으며, 복구 전에는 f1 스코어가 0.2대에서 맴돌던 것이 여러 노력 끝에 상위 점수대로 올랐다는 사실만으로도 큰 의미가 있었다.

지금은 비록 이번 점수가 아쉬울 수 있지만, 그 과정에서 배운 경험들은 다음 프로젝트에서의 비상을 위한 밑거름이 될 것이다. 스코어 표를 보면 아쉽게도 우리가 꿈꾸던 그 이상에 도달하지는 못했지만, 다시금 목표를 향해 도전할 수 있는 힘을 얻었다고 믿는다. 오늘의 땅에 묻은 씨앗은 내일의 열매로 돌아올 것이라 확신하며, 더 나은 결과를 향해 나아갈 다짐을 남긴다.

언젠가 "내일의 우리"는 오늘을 추억하며 미소 짓고 있겠죠.

precision	recall	f1	제출횟수	최종 제출
0.8144	0.8506	0.8321	27	1w

Table 2: final score