

CV-08 랩업 리포트

-Level-1 Image Classification 대회 리뷰-
CV-08조 팀 SEE-Beyond-Accuracy

1. Project outline

스케치 데이터의 특성을 이해하고 스케치 이미지를 통해 모델이 객체의 기본적인 형태와 구조를 학습하고 인식하도록 함으로써, 일반적인 이미지 데이터와의 차이점을 이해하고 또 다른 관점에 대한 모델 개발 역량을 높이는 데 초점을 두었습니다.

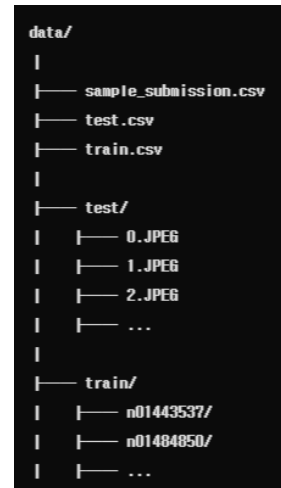
이를 통해 실제 세계의 복잡하고 다양한 이미지 데이터에 대한 창의적인 접근방법과 처리 능력을 높일 수 있습니다. 또한, 스케치 데이터를 활용하는 인공지능 모델은 디지털 예술, 게임 개발, 교육 콘텐츠 생성 등 다양한 분야에서 응용될 수 있습니다.

제공되는 이미지는 주로 사람의 손으로 그려진 드로잉이나 스케치로 구성되어 있습니다. 객체, 동물, 인물, 풍경 등 다양한 카테고리에 걸친 추상적이고 간소화된 이미지들로 이루어져 있으며, 색상이나 세부적인 질감 없이 기본적인 윤곽선과 형태로 표현됩니다. 같은 객체를 나타내는 스케치라도 다양한 시각과 표현 스타일을 반영하기에 그린 사람에 따라 매우 다를 수 있습니다.

평가 지표는 정확도 (Accuracy)를 사용합니다. 정확도는 전체 예측 중에서 맞게 예측한 비율을 측정하여, 모델이 실제 객체(class)를 얼마나 잘 맞추는지를 평가합니다.

프로젝트 전체 기간 (2주) : 9월 10일 (화) 10:00
~ 9월 26일 (목) 19:00

※데이터셋 구조도



2. Team Members and Roles

Name	Role
임용섭	EDA, Hyper Parameter, Augmentation
박재우	EDA, 데이터 전처리, Modeling
이상진	EDA, 데이터 전처리, Augmentation, refactoring
정지훈	EDA, 데이터 전처리, Modeling
천유동	EDA, 데이터 전처리, Modeling
유희석	EDA, 데이터 전처리, Augmentation, Modeling

✓ Team Project Target

- Leader Board 성능 보다는 Robust한 모델을 위한 설계를 해보자
- 다양한 모델과 실험 가설들을 실험해보자.

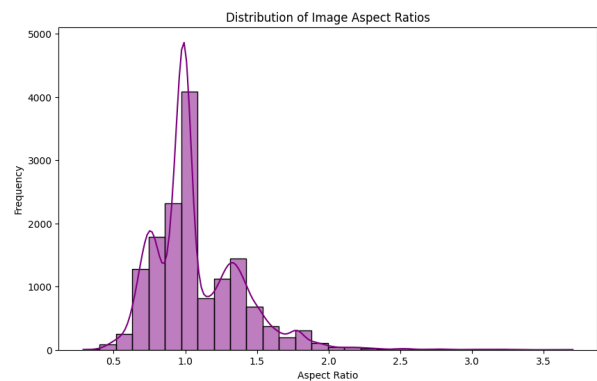
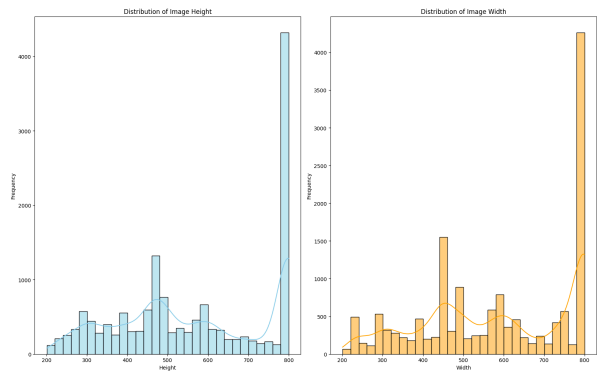
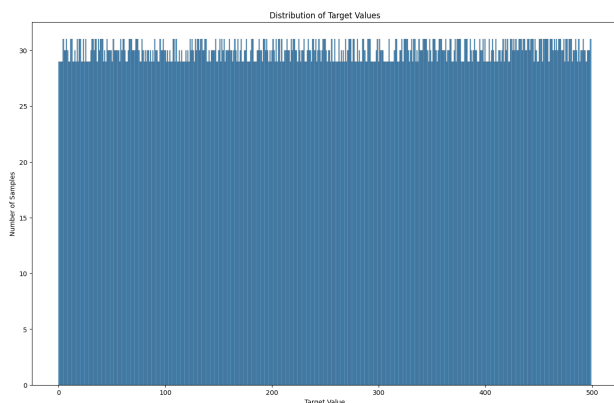
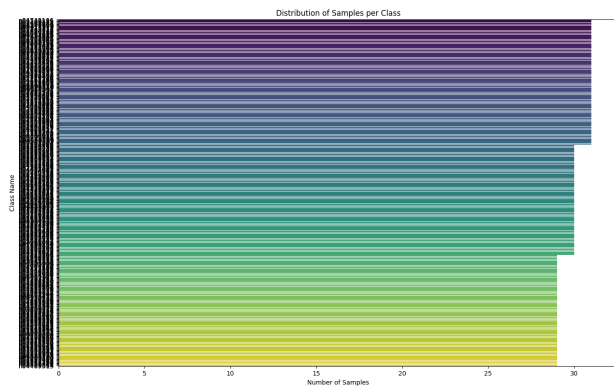
3. Project

3.1 EDA

(1) EDA

✓ Label

각 클래스가 모두 Data 29~31개로 밸런스 잡힌 상태를 이루고 있음

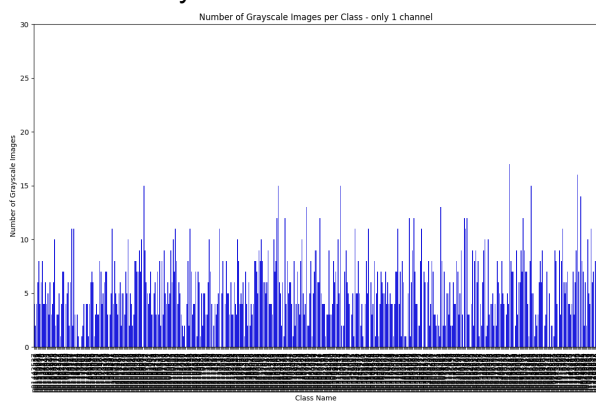


✓ RGB

대부분 RGB 값이 200~250 사이에 분포해서 전반적으로 이미지의 밝기와 채도가 높은 편

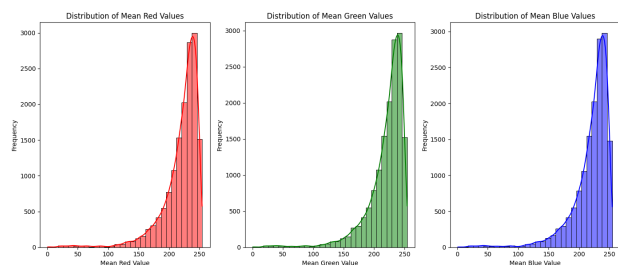
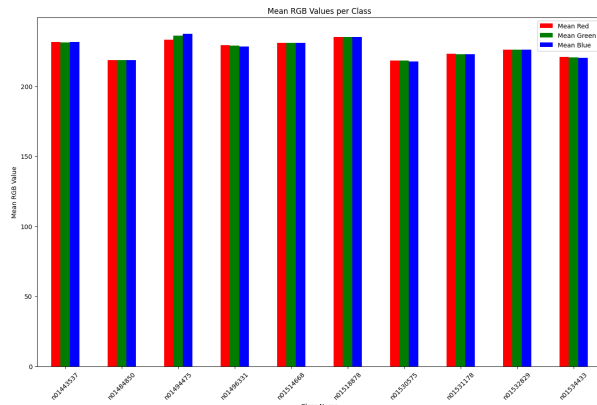
✓ GrayScale

RGB로 이루어진 데이터 사이에 1채널로만 이루어진 Grayscale 데이터가 있음



✓ Size

이미지가 일정한 크기가 아니라 200~800 픽셀까지 다양하게 분포되어 있고 가로세로 비율도 다양함



♂ Approach :

- 1) 전처리 & 데이터 증강을 이용하여 해결해 보자
- 2) Train/Validation set을 균형 있게 구성해 보자

✓ Background

이미지 데이터에서 여러 가지 클래스의 사물이 같이 있거나 다른 클래스의 그림이 섞여 있음.

- 🏃♂️ Approach :

- 1) 모델에 CenterCrop이나 RandomCrop을 통해서 해결해볼 수 있을 것 같다
- 2) 불필요한 이미지 제거를 해야 할 것으로 보인다

3.2 Data

(1) Data Preprocessing

✅ RGB & GrayScale

모든 데이터를 RGB로 만들거나 GrayScale로 만들어서 학습

- ★ Convert RGB or GrayScale

장점 : 데이터의 일관성 증가
단점 : 학습 이외에 추가적인 시간 소요

Method	Loss	Accuracy
RGB	0.118	0.842
GrayScale	0.1157	0.841

RGB와 GrayScale 모두 비슷한 성능을 보임

✅ Train / Validation Set

- 'Random Split' : 모든 데이터를 랜덤적으로 Split -

실험할 때마다 Validation set이 달라질 수 있음

- 'Split with Seed' : Seed 기준으로 데이터를 Split -

여전히 서버마다 랜덤하게 Validation set이 정해짐

- Validation data set을 어떤 데이터로 할 지 사전에 결정 :

Validation set과 Train set을 나누는 기준을 사전에 하나의 파일로 데이터 목록을 설정해 나누도록 해서 위의 두 문제를 어느정도 해결

(2) Augmentation

✅ Rotation & Flip

클래스 별로 30개 정도 밖에 데이터가 없는 상황을 해결하기 위해서 기본적으로 Rotation과 HorizontalFlip을 진행하여서 데이터를 증폭 시켜 줌

Method	Loss	Accuracy
Basic Augmentation	0.836	-
Rotation + Flip	0.7875	0.794

✅ Rotation & Flip + a

Rotation과 HorizontalFlip에 더해서 Affine, Jitter, Crop등을 진행하여서 데이터를 증폭 시켜 줌

Method	Loss	Accuracy
Rotation + Flip	0.7875	0.794
Rotation + Flip + a	1.2219	0.69

너무 많은 Augmentation기법을 한번에 사용하면 오히려 악영향을 미침

✅ Mix up

추가로 Mix up을 진행하여서 결과를 실험함

Method	Loss	Accuracy
Basic Augmentation	0.7372	0.811
Use Mix-up data set	0.846	0.794

Loss가 오히려 상승해서 사용하지 않는 것으로 결정

✅ Cut mix

과적합을 피하기 위해서 Cut mix를 추가적으로 활용

Validation Score와 Test Score 증가

Method	Loss	Accuracy
Basic Augmentation	0.8593	-
Cut mix	0.8084	0.824

3.3 Modeling

(1) Modeling

✅ Hyper Parameter

Loss, Optimizer, Lr는 경험적 실험 결과, 비슷한 성능을 보여서 아래와 같이 고정

- Loss : CrossEntropyLoss
- Optimizer : AdamW
- Lr : 0.0005

✅ Base Model : ResNet152

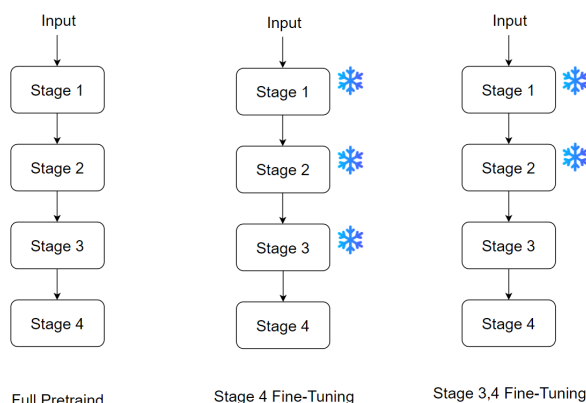
Why ResNet152 : Transformer 계열의 경우 많은

양의 데이터가 필요하다. 하지만 우리가 갖고 있는 데이터의 수는 충분하지 않다. 실험을 해 본 결과 **Pretrained** 된 모델의 가중치를 가져와도 우리가 갖고 있는 데이터와 **Pretrained**에 활용되는 데이터의 유사도가 적기 때문에 **Transformer** 계열은 **CNN** 모델에 비해 유의미한 결과를 내지 못했다. 또한 서비스를 제공하는 관점에서도 **Transformer** 계열의 경우 학습/추론에 많은 시간이 소요되기 때문에 효과적인 학습과 적은 데이터에서도 괜찮은 성능을 내기 위해서 **CNN** 계열의 모델로 방향을 잡았다.

CNN 계열의 **Efficientnet**, **Resnet**, **MobilenetV2** 등을 비교한 결과 **Resnet**이 꾸준히 좋은 성능을 보여서 **Resnet** 계열을 선택하였다

Method	Loss
efficientnet_b3	0.853
resnetrs50	1.6818
mobilenetv2	1.6356
resnet152	0.7316
deit_small	1.19253

✓ Fine-Tuning



ResNet152 : 사전 학습된 **ResNet152** 모델을 프로젝트에 맞게 **fine-tuning**하였으며, 4번 **stage**와 3, 4번 **stage**를 재학습하여 각각의 방법이 모델 성능에 미치는 영향을 실험하였다.

Method	Loss
--------	------

Full Pretrained	0.7996
Stage 4 Fine-Tuning	1.08242
Stage 3,4 Fine-Tuning	0.81792

의외의 결과로 3,4번 **Stage**를 재학습한 경우가 전체 **Pretrained** 모델보다 **Validation Loss**가 낮게 나왔다. 모델의 특정 레이어를 재학습함으로써 성능 개선이 기대되었으나, 오히려 전체 모델을 재학습하는 것이 효과적일 수 있음을 보여준다.

(2) Ensemble

✓ Test Score와 모델의 다양성을 기준 Ensemble

Accuracy와 모델의 다양성을 기준으로 상위 모델을 선택해서 **Hard Voting** 및 **Soft Voting** 두 가지 방식으로 진행했다.

Hard Voting의 경우 여러 모델들의 **output csv**파일들의 **target**값이 가장 많이 나온 **target**을 선정하여 결론을 도출했다.

Soft Voting의 경우 여러 모델들의 **output**을 **softmax** 함수를 통해 확률 값으로 반환하여 평균치를 계산해 결론을 도출했다.

1. 정확도 기반 Hard Voting

- Deit small
- Resnet 152 + CLAHE data
- Resnet 152 + Auto Augmentation (CIFAR10) Grayscale data
- Resnet 152 + Auto Augmentation (IMAGENET) RGB data
- Resnet 152 + shear data
- Resnet 152 + translate data
- Resnet 152 + 중복데이터 제거 data + Focal Loss

2. 정확도 기반 Hard Voting

- Hard Voting 1번 모델들
- Resnet 101 + cutmix + 원본 + flip + rotation + jitter + affine + blur + auto
- Resnet 152 + cutmix data
- Resnet 152 + cutmix + 원본 + flip + rotation + jitter + affine + blur + auto + drop out 0.3
- Resnet 152 + Auto Augmentation (IMAGENET) RGB data + more validation data

3. 정확도 기반 Hard Voting

- Hard Voting 2번 모델들
- Resnet 152 + SGD + cutmix data
- Resnet 152 + SGD + cutmix data + drop block
- Resnet 152 + SGD + cutmix data + drop path

4. 모델 다양성 기반 **Hard Voting**

- Resnet 50
- DeiT
- Swin Transformer
- Resnet 101
- Resnet 152
- Resnet 152 + cutmix + 원본 + flip + rotation + jitter + affine + blur + auto
- Resnet 152 + Focal loss
- Regnety 080

5. 모델 다양성 기반 **Soft Voting**

- 위와 동일

✓결과 : (Test Score 기준)

Method	Accuracy
Ensemble 1	0.8720
Ensemble 2	0.8880
Ensemble 3	0.8880
Ensemble 4	0.8900
Ensemble 5	0.8940

Ensemble 5가 가장 높은 정확도를 보였다

가장 좋은 Hard Voting 모델들로 Soft Voting을 진행하여 더 일반화된 모델이 만들어지지 않았나 추론

4. Final Score

✓최종결과

- Public Score

22 CV_08조  0.8920

- Private Score

22 CV_08조  0.8940

Public Score에 비해서 Private Score가 소폭 증가하였지만

최종순위 **22위** 기록

5. Cooperation

A. 협업 방법

- Google Spreadsheet을 통한 프로젝트 진행 과정 및 실험 결과를 공유
- Git과 Github를 활용하여서 개인이 작성한 코드를 공유하고 서로 리뷰
- Zoom을 활용하여 실시간으로 소통하면서 협업진행

B. 개발 환경 및 Tool

- 개발언어 : Python
- 개발환경 : V100 32GB GPU
- Frame Work : Pytorch
- 협업툴 : Slack, Notion, Git

6. 팀 회고 및 개선방안

✓잘했던 점

- 팀원들끼리 자유로운 아이디어의 공유가 좋았다
- 대회에 익숙하지 않은 팀원들도 잘 적응할 수 있었다
- 가설을 세우고 이를 검증하면서 모델링을 진행한 것이 좋았다
- 회의를 통해서 역할을 분담하고 체계적으로 진행한 것이 좋았다

✓아쉬웠던 점

- 개인마다 진행 속도가 차이가 나서 아쉬웠다
- Git과 같은 협업툴을 적극적으로 활용하지 못해서 아쉬웠다
- 실험 공유에 Convention을 정하지 못해서 아쉬웠다
- Python Code 버전 관리를 하지 못해서 동일한 조건을 맞추기 힘들었고 새로 구현한 기능의 호환성을 맞추기 힘들었다

7. 개인 회고

🤖 천유동

- 프로젝트 학습 목표

- 인공지능에 관심있는 사람들로 이루어진 첫 프로젝트였기 때문에 프로젝트 진행을 원활하게 하는 것이 첫 목표였다. 어떤 방식으로 진행이 되고 지금까지 배운 기초 지식들과 이론들을 사용하여 적용하려고 노력했다.

2. 성능도 중요하지만 개인적인 이론이나 가설의 검증을 통해 스케치 데이터 분류를 이해하는 것이 중요하다고 생각했다.

- 내 학습 목표를 달성하기 위해 한일

1. EDA

- 우선 EDA 코드를 실행하기 직전에 학습 데이터 15000개, 테스트 데이터 10000개, 500개의 클래스와 각자 고르게 분포된 것을 확인했다. EDA 코드를 실행하며 분석한 결과 데이터 자체는 흑백사진일 수 있지만 기본적으로 RGB로 이루어진 형태가 많았으며 Grayscale로 이루어진 데이터가 일부 있었다. 추후 코드에서 RGB와 Grayscale 모두 진행하며 둘 중 어느 방식으로 진행하는 것이 효과적일지 고민을 하였다. 또한 새로운 코드를 작성하여 해시값의 비교를 통해 중복데이터의 존재를 확인하였다. 육안으로도 확인을 하여 일부 데이터 속 객체들이 중복된 현상, 대비가 명확하지 않은 현상 등을 발견하여 데이터 전처리를 위한 토대를 세웠다.

2. 모델 설정

- 모델 설정의 경우 여러가지 방법이 있었지만 일단 하이퍼파라미터들을 고정된 이후 각종 모델을 돌려가면서 어느 모델이 성능이 좋은지 확인을 하였다. 또한 데이터의 개수가 500개 클래스에 30개 가량의 데이터 밖에 없었기 때문에 CNN을 트랜스포머 기반 모델들보다 더 선호했다.

3. 하이퍼 파라미터 설정

- 모델 설정과 비슷하게 나머지 값들은 시드값 고정 및 각종 변수 고정을 한 상태로 하나의 변수만 비교해가면서 어느 매개변수가 좋은 값인지에 대해 비교하였다.

4. Augmentation 실험

- 데이터 증강의 경우, CNN 기반 모델들보다 트랜스포머 기반 모델들에 한해서 데이터가 부족하다고 판단하여 전부 진행하였다. 기본적으로 자동증강을 진행하였으며 RGB용 auto augmentation은 IMAGENET 정책으로 진행하였으며, Grayscale의 경우 색도조정이 필요 없다고 생각하여 CIFAR10정책을 활용해서 적용하였다.

5. 앙상블

- 마지막에는 csv파일을 기반으로 hard voting을 하며 앙상블을 진행하였고 꽤나 효과가 있었다.

- 나는 어떤 방식으로 모델을 개선했는가?

배치사이즈 조절을 통해 메모리가 부족하지 않은 선에서 최적의 배치사이즈를 찾았다. 또한 손실함수들의 변수들도 조정하며 최적의 파라미터들을 찾았다. 베이스라인 코드에서 시드값 고정을 시키며 결과값을 일정하게 비교할 수 있도록 수정하였다.

- 학습 목표를 달성하기 위한 과정 중 깨달은 점

프로젝트를 진행하며 깨달게 된 점은 나의 개인적인 이론과 실제 결과값이 다른 부분이 꽤나 많았다. 실험을

진행하며 SGD가 Adam보다 수렴이 늦다는 것은 이론적으로 알았지만 오히려 Adam보다 정확도 자체는 더 높게 나오는 경향이 있었다. 또한 CNN 기반 모델들이 트랜스포머 기반 모델들보다 성능이 좋게 나오는 등 개인적인 생각과는 다른 부분들이 있었다. 또한 배치크기가 클수록 좋았고, 일부 데이터 증강의 효과는 보았다는 점에서 이번 프로젝트에 한해서 해당 작업들의 중요성을 알 수 있었다.

- 아쉬웠던 점

1. 프로젝트를 상대적으로 다른 팀들보다 늦게 시작하여 시간과 제출횟수가 상대적으로 부족하였다.

2. 위의 문제로 인해 시간이 부족해서 각자 급해져서 굉장히 정돈되지 않은 상태로 프로젝트를 진행하였다. 코드 통일, 시드값 통일, 검증 데이터 고정, EDA등 조 내에서 통일시키고 진행하지 못하여 서로 비교 시 비교가 제대로 되지 못했다. 나중에는 통일하였지만 조금 늦었다.

3. 개인적으로는 git활용이 아쉬웠다. 늦게 시작한 것과는 별개로 github를 통한 협업이 이번이 처음이었기 때문에 내가 너무 못했다.

4. 후반부에 주로 CNN을 기반으로 실험들을 진행하였는데, 나는 데이터 증강 이후, 트랜스포머 기반 모델들을 시도하며 여러가지 실험을 진행하였지만 결국에 결과값이 좋지 못해서 팀에 딱히 도움이 되지 못한 것 같아서 미안했다. 다른 조의 경우 트랜스포머도 활용한 것을 보면 내 기본기가 부족해서 그렇게 된 것 같다.

5. 특정 이미지 클래스들이 분류가 어려울 것이라고 판단하여 triplet loss를 사용한 contrastive learning을 적용시켜 resnet 152 마지막 레이어에 각자 클래스별 특징을 담은 레이어를 추가하여 정확도를 개선하려 했지만 개선시키지 못해서 아쉬웠다.

- 총평

아쉬운 부분이 많은 첫 프로젝트였지만 이번 프로젝트를 통해 얻은 점은 굉장히 많았다. 기본적으로 프로젝트 진행 시 어떤 방법으로 진행되어야 하는지와 팀끼리 협업, 기록의 중요성, 그리고 여러가지 CV기반 경험을 얻게 되어서 만족스러웠다.

😊 임용섭

- 프로젝트 학습 목표

1. 인공지능 분야에서 처음 경험하는 협업 프로젝트였기 때문에 활발한 의사소통을 통해 팀원들의 의견을 조율하고 효율적인 협업을 통해 문제를 해결하며 프로젝트를 수행하면서 앞으로 있을 협업 활동에도 익숙해 질 수 있도록 하는 것이 목표였다.

2. 주어진 문제에 대해 문제를 정의하고 배웠던 이론적인 지식을 기반으로 세운 가설을 세우고 실험을 통해 검증하는 경험을 하여 많은 인사이트를 얻는 것이 개인적인 학습 목표였다

- 나는 내 학습 목표를 달성하기 위해 무엇을 어떻게 했는가?

1. EDA 결과를 통해 가설을 세우고 가설을 검증하기 위해서 실험을 시도하였다.

2. 팀원들과의 소통을 통해 나의 가설과 결과를 공유하였고, 팀원들의 결과를 종합하여 다음 가설 및 실험을 진행하였다.

- 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

1. 프로젝트 후반부에 **Score**를 올리고자 집착한 것 같고, 부족한 시간을 활용하기 위해 체계적인 실험을 하지 못했다. 미리 시간을 효율적으로 활용할 수 있는 도구와 방법을 이용해 많은 실험을 해 놓았어야 했다고 생각했다.

2. 시간분배를 잘못하여 시간이 부족하다보니 가설을 세우고 진행하기보단 그냥 아무거나 돌아보이는 걸 해보자 이런 식으로 진행이 되어서 결과가 나와도 분석을 하기 어려웠다. 다음 프로젝트는 가이드 라인을 정하여서 가이드 라인을 준수하며 프로젝트를 진행하면 좋을 것 같다.

3. 6명의 팀원과 4개의 서버로 프로젝트를 진행하다보니 서버를 사용하고 있으면 실험을 못하는 경우가 있어서 기다리는 시간이 아쉬웠다

- 나는 어떤 방식으로 모델을 개선했는가?

1. 모델이 너무 깊어서 학습을 잘 못한다고 생각하여 **drop_path** 사용

2. 이미지의 3채널과 1채널 비교를 통해 모델의 성능 비교를 통해 모델의 인풋 결정

3. **AutoAugmentation** 데이터의 효과를 확인하기 위해 실험 진행

- 내가 해본 시도 중 어떠한 실패를 경험했는가? 실패의 과정에서 어떠한 교훈을 얻었는가?

1. **overfitting** 문제는 **metric**과 **loss**만으로 구별하기가 어려웠기 때문에 신뢰도 있는 **validation set**을 구성하던가 혹은 따로 **test set**을 만들어서 검증하는 방법을 활용해야겠다고 생각했다.

2. 서버마다 같이 사용하는 **validation set** 구성이 필요하다는 생각이 들었다.

3.. 프로젝트 후반부에 부족한 시간에 **score**를 올리고자 **ablation study** 방식으로 실험을 진행하지 못하였고 이로 인하여 실험 내용 정리가 어렵고 명확한 인사이트를 얻기가 굉장히 힘들어 졌다. 이전 실험 내용들을 미리 정리를 해 놓지 않은 상태에서 많은 실험이 쌓이면서 최종 모델을 선택하는 것이 어려웠다.

- 협업 과정에서 잘된 점/ 아쉬웠던 점은 어떤 점이 있는가?

잘랐던 점

• 가설이나 실험 결과에 대해 공유하고 토론해 보면서 다양한 관점에서의 문제를 해결하려는 생각들을 알게 될 수 있어서 좋았다.

• 혼자 해결하기 어려웠던 코딩 이슈나 사용해보기 못했던 방법론에 대해 알려줘서 시간이 오래걸릴 수 있는 문제를 빠르게 해결할 수 있어서 좋았다.

아쉬운 점

• 팀원 간 의사소통은 활발하게 되었지만 실험 내용, 코드를 정리하여 공유하는 것이 잘 되지 않았다. 따라서 프로젝트가 진행 될 수록 개인간의 격차가 커져서 프로젝트 진행 상황을 하나로 모으기 힘들었다.


- 한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

• 프로젝트 진행 전 충분한 회의와 의사 소통을 통해 팀의 목표를 설정할 것이다.

• 클린 코드를 만들고 협업 도구를 많이 활용하여 팀원들과 같이 프로젝트를 진행할 것이다.

• 실험 내용을 미리 정리하고 공유하여 다양한 관점을 통해 앞으로의 실험 내용을 정할 것이다.

• 리더보드 순위에 연연하지 않고 효율적인 프로젝트 진행을 통해 컨디션 관리를 할 것이다.

 정지훈

- 나는 내 학습 목표를 달성하기 위해 무엇을 어떻게 했는가?

• 처음 경험해보는 대회이므로 프로젝트의 전체적인 흐름과 협업의 과정에 대해 배워야겠다고 생각했다.

• **baseline code**에 대한 전반적인 이해를 확실하게 하기 위해 코드를 열심히 뜯어봤다.

• 전반적인 데이터에 대한 이해를 위해서 모든 클래스의 이미지를 한번씩 들여다보고 통계를 내보았다.

- 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

• 커뮤니케이션은 원만했으나 생각보다 협업이 원활하게 이루어지지 못했던 것 같다. 서버로 나뉘어서 파일 공유와 백업도 원활하지 못했고 협업 툴도 많이 사용하지 않았다.

• 내가 직접 코드를 짜기 힘들어서 **gpt**에 의존을 많이 한 것 같다. 그리고 그 코드에 대해서 깊은 이해를 못 한 것이 아쉽다.

• 모델에 대해 많이 알지 못해 어떤 모델을 실험해봐야 할지와 어떤 모델로 선택할 지에 어려움이 있었다.

• 결과를 제출한 후에야 정확도를 알 수 있어서 **output**을 제출하지 않고도 대략적으로라도 모델의 성능을 확인할 수 있는 지표를 설정하는 게 필요해 보인다.

- 한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

- 여러 가지 모델에 대해서 알아보고 **zero-shot**으로 많이 실험해보고 결정하면 더 나은 성능을 뽑아낼 수 있을 것 같다.

- 다음 프로젝트때는 깃허브를 적극적으로 활용해서 협업, 백업, 버전 관리를 제대로 해보고 싶다.

- 프로젝트 초기에 협업 툴의 규칙을 정확하게 만들어놓자

- 사람마다 속도가 다 다르므로 나의 페이스를 지키는 것을 목표로 하자.

- 내가 해본 시도 중 어떠한 실패를 경험했는가?
실패의 과정에서 어떠한 교훈을 얻었는가?

- 여러 번의 실험을 했지만 실험 결과가 자꾸 들쭉날쭉하고 왜인지 모를 이유로 아예 학습이 안되어서 **0.002**정도가 나오는 경우가 있었다.

- **validation set**과 여러 가지 랜덤성이 있는 부분을 통제하지 못한 부분이 문제였던 것 같다. 시드를 고정하고 여러 번의 실험을 통해서 평균값을 구해서 해결하면 될 것 같다. 이유를 알 수 없는 낮은 정확도를 보이는 경우에 대해서는 무시한 후 중단하고 다시 학습을 시도해보면 제대로 나오는 경우가 많아서 여러 번 돌려보는게 가장 중요한 것 같다.

- **Grad-CAM**을 통해서 이미지의 어디에 주목하는지와 어떤 클래스가 제대로 된 부분을 보지 못하고 있는지를 보고자 코드를 설계해 보았다.

- 그런데 자꾸 메모리 부족과 이미지 인식 불가 등의 에러로 인해서 제대로 된 구현에 실패했다.

- 이를 통해 코딩 실력이 아직 많이 부족하다는 점과, **gpt**가 말해준 정보들과 코드에 대해서 내가 제대로 이해할 수 있는 정도만이라도 실력을 키워야겠다고 생각했다.

- 협업 과정에서 잘된 점/ 아쉬웠던 점은 어떤 점이 있는가?

- 아이디어를 제시하고 받아들이는 데에 거리낌이 없었다는 점이 팀원 모두가 협업에 적극적이었다.

- **Google Spreadsheet**를 통해서 실험 결과를 공유하고 기록하는 부분은 잘 된 것 같다.

- 하지만 **github**, **notion** 등 협업 툴을 통한 실험 진행, 코드 등에 대해 공유가 많지 않아서 팀원들이 어떤 실험을 어느 정도까지 진행을 했는지, 앞으로 해야할 실험은 무엇인지가 체계적이지 않아 실시간 피드백이 어려워 팀으로써의 장점이 퇴색된 것 같다.

- **github**를 통해서 버전 관리를 하면서 코드를 서로 볼 수 있도록 진행하면 훨씬 원활한 협업을 진행할 수 있을 것 같다.

😊이상진

- 개인적 감상

- 학습 환경이 낯설고, 너무 급했다.

- 코드의 완성도와 스코어에 집착했던 것 같다

- **EDA**과정에서 아쉬운점이 많다

- 체계적인 실험관리가 필요하다 느낀다.

- 다음 프로젝트에서는 이번의 실수를 하면 안되겠다.

- 프로젝트 학습 목표

1. 하나의 **python**파일로 여러가지 실험을 진행할 수 있을까 고민하였다.

2. 프로젝트 실험을 체계적으로 관리하고자 하였다.

3. 인공지능 프로젝트를 어떤식으로 진행되는지 전반적인 프로세스를 이해하고자 하였다.

- 나는 내 학습 목표를 달성하기 위해 무엇을 어떻게 했는가?

- **train, test, main.py**로 나눠 학습만, 테스트만, 학습부터 테스트 까지 하는 기능을 만들었다.

- **Notion**에서 간트차트와 같은 세부적인 프로젝트 페이지 개설

- 스프레드 시트에 최대한 자세히, 어떤 실험을 하였는가 적어 놓았다

- 베이스 라인 코드를 뜯어보고, 이해하는 식으로 진행하였다.

- 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 실험 스프레드 시트를 관리하는건 좋았지만, 여러사람이 함께 사용하고 실험결과에 대한 소통이 잘되지 않았다.

- **Notion**페이지 간트차트는 거의 사용되지 않았다.

- **train, test, main.py**로 만들어 각각 다른 기능을 하도록 프로젝트가 끝나고 수정하였으며, **inference.py**를 만들어 이미지를 **input**으로 넣어주면 **class**를 구분하는 파일을 만드려 했지만 수행하지 못하였다.

- **config.py**에서 클래스로 모든 하이퍼 파라미터를 관리하는데 더욱 파일이 복잡해 지는 문제가 있었다.

- **git**에서 버전관리가 안되어 서로 실험 환경이 달라지는 문제가 있었다.

- 한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

- 스코어에 집착하지말고, **EDA**에서 나온 가설을 검증하고 검증에 따른 결과에 힘을 주자

- 처음 프로젝트 시작 시 리팩토링 후 **github**에 **push**하여 개인별로 브랜치를 열어 실험을 하는 방식으로 진행

- **streamlit**을 이용해 **input**에 이미지를 넣으면 **detection**된 결과를 확인할 수 있도록 만들어보자

- **wandb**를 이용한 실험결과 관리

• K-Fold를 이용한 Validation Set Test

- 나는 어떤 방식으로 모델을 개선했는가?

- 다양한 실험을 하기 위해 하이퍼 파라미터를 `config.yaml`으로 관리하도록 하였다.
- **ResNet Fine-tuning**을 하여 3,4번 레이어를 동결 후 재학습 하는 실험을 진행하였다
- `train,test.py`로 만들어 단일 프로세스로 진행하도록 하였다.

- 내가 해본 시도 중 어떠한 실패를 경험했는가? 실패의 과정에서 어떠한 교훈을 얻었는가?

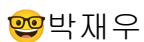
- 서로 다른 실험 환경을 하나의 프로젝트로 합치는 과정이 너무 어렵다 느꼈다.
- **EDA**를 더 깊게 파서 어느부분에서 수정하게되면 더 좋은 결과를 얻을 수 있는 지 파악이 필요하다.
- 실험은 철저한 계획을 수립한 이후 진행하고, 실험관리를 제대로 하는게 좋겠다.

- 협업 과정에서 잘된 점/ 아쉬웠던 점은 어떤 점이 있는가?

- 아이디어 도출과 아이디어를 받아드림에 있어 유연하게 진행되었다.
- 프로젝트를 진행함에 있어 각자의 포지션이 없어서 아쉬웠다.
- 서로간 결과에 대한 소통이 부족함을 느꼈다.

- 팀 내 자신의 역할, 프로젝트에서 사용한 자신의 기술 및 지식, 모델링 및 성능 개선 등 프로젝트 전 과정에서 자신이 기여한 내용과 개인의 구체적 성과를 정리

- 베이스라인코드를 참고하여 리팩토링을 진행하고, 최종 **project**의 틀을 잡아냈다.
- **Mixed Precision, CutMix** 등 파일을 리팩토링하여 재사용 가능하도록 만들어냈다.
- ViT기반 모델들을 시험하고, **Overfitting**에 문제가 있음을 제시하여 **CNN**기반으로 가자 제안



박재우

- 느낀점

- 이번 프로젝트를 통해 한달이라는 시간동안 배웠던 다양한 딥러닝 기법과 모델링 방법론을 실제로 적용해볼 수 있어서 아주 값진 경험이었다. 특히, 스케치 이미지라는 일반적인 이미지와 다른 특성을 가진 데이터를 다루면서

데이터 전처리, 모델 구성, 성능 개선 등을 직접 실험하고 분석해보는 과정에서 많은 인사이트를 얻게되었다.

-나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

• EDA

EDA 단계에서는 전체 **train** 데이터를 하나씩 면밀히 살펴보고 다양한 특징들을 파악하였다. 이를 통해 각 클래스의 대략적인 분포를 확인할 수 있었고, 일부 클래스에서는 데이터가 중복되어 있는 현상을 발견하게 되었다. 또한, 한 이미지에 여러 객체가 포함된 사례와 서로 다른 클래스임에도 동일한 이미지를 공유하고 있는 데이터도 존재한다는 점을 확인하였다.

• 모델 탐색

스케치 이미지 분석에 적합한 모델을 찾기 위해 **timm** 라이브러리에 포함된 다양한 **CNN** 기반 모델들을 폭넓게 실험해 보았다. **ResNet, RegNet, EfficientNet** 등의 대표적인 모델을 활용하여 성능을 비교 분석하였으며, 각 모델의 **train loss, validation loss, accuracy**, 그리고 학습 시간 등 주요 지표들을 체계적으로 기록하였다. 이러한 결과를 바탕으로, 각 모델의 장단점을 파악하고 프로젝트 목표에 가장 부합하는 모델을 선정하기 위해 팀원들과 긴밀하게 논의하였으며, 실험 시트를 통해 성능 결과를 서로서로 시각적으로 공유했다.

• 손실 함수 탐색

Baseline 코드에서 기본적으로 제공되는 **Cross-Entropy Loss** 대신, 우리의 **train** 데이터에 더욱 적합한 손실 함수를 찾기 위해 다양한 실험을 진행하였다. 먼저, 여러 논문과 자료들을 조사한 결과 **Label Smoothing Loss, Softmax Loss, Focal Loss, Asymmetric Loss** 등의 손실 함수가 스케치 이미지 분류에 효과적일 수 있다는 점을 확인하게 되었다. 이에 따라, 각 손실 함수를 구현하여 모델 학습에 적용해 본 후, 성능 지표를 기반으로 비교 분석을 수행하였다. 각 손실 함수들도 모델 분석때와 마찬가지로 **train loss, validation loss, accuracy** 등을 측정하여 성능을 평가하였다.

• 데이터 증강 실험

EDA 결과, 각 클래스별 **train** 데이터가 약 30장 정도에 불과한 매우 소규모 데이터셋이라는 점을 확인하게 되었다. 데이터의 양이 충분하지 않으면 모델이 과적합에 빠지기 쉽고, 일반화 성능이 저하될 수 있기 때문에, 다양한 데이터 증강 기법을 통해 데이터를 효과적으로 늘려주는 전략이 필요하다고 판단하였다. 이에 따라, 자동 데이터 증강 기법인 **AutoAugmentation**을 비롯하여 **CutMix, Flip, Rotation** 등의 다양한 증강 방법을 적용해 보았다. 각 기법이 모델의 성능에 미치는 영향을 확인하기 위해, 각기 다른 증강 조합들을 실험적으로 적용해 성능 지표를 비교 하였고, 최적의 증강 조합을 찾기 위해 수많은 반복 테스트를 진행했다. 이러한 실험 과정을 통해 모델의 학습 데이터 다양성을 확보할 수 있었다.

-마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 코드 구현 능력의 부족

프로젝트 초기 단계에서 **baseline** 코드 자체를 이해하는 데에 많은 시행착오가 있었다. 기존 코드의 구조와 흐름을 파악하는 데 시간이 소요되었고, 이를 완벽하게 이해하지 못한 상태에서 새로운 기능을 추가하거나 커스터마이징하려다 보니 구현 과정에서 크고 작은 문제들이 빈번히 발생했다. 특히, 새로운 손실 함수를 적용하고자 할 때, 여러 논문 및 관련 자료에서 제공되는 **base** 코드를 우리 프로젝트의 코드베이스에 맞게 통합하는 과정에서 수많은 오류에 직면하게 되었다. 이때 발생하는 오류들은 단순한 문법적인 문제가 아니라, 코드의 논리적인 흐름이나 데이터 타입 간 불일치, 라이브러리 간의 호환성 이슈와 같은 복잡한 문제들이 많았기에 이를 해결하는 데 많은 시간을 할애했다. 하지만 결국 몇몇 손실 함수들은 이러한 오류를 끝내 해결하지 못해 프로젝트에 적용할 수 없었다.

• 깃 활용의 아쉬움

프로젝트를 진행하며 협업 도구로써 깃을 사용했지만, 그 활용도가 기대에 미치지 못해 아쉬움이 많이 남는다. 깃을 단순히 코드 저장소로만 사용한 느낌이 강했고, 실제 협업 도구로서의 강력한 기능들을 충분히 활용하지 못한 것 같다. 특히, 팀원들과의 원활한 코드 병합 및 리뷰, 그리고 이슈 관리 등을 깃을 통해 체계적으로 수행하기보다는, 개인이 수정한 코드를 단순히 업로드하고 저장하는 용도로만 깃을 활용한 듯 하다. 그리고 **Visual Studio Code**와 깃을 연동하여 사용하는 데에도 익숙하지 않아 작업 속도가 느려지고 불필요한 시행착오 또한 경험하였다.

• 가설 및 검증의 어려움

모델의 성능을 향상시키기 위해 여러 가지 가설을 세우고 이를 실험적으로 검증하는 과정에서 많은 어려움이 있었다. 특히, 각 가설을 바탕으로 실험을 수행한 후, 그 결과가 기대한 바와 일치하지 않거나, 성능 지표 간의 차이가 미미해 유의미한 결론을 도출하기 어려웠을 때, 문제의 원인을 파악하고 대안을 모색하는 데 큰 어려움을 느꼈다. 여러 가설을 세워 실험을 반복했지만, 일부 실험에서는 가설과 정반대의 결과가 도출되거나, 데이터의 특성으로 인해 예측할 수 없는 결과가 발생하여 당황스러운 순간이 다소 있었다.

또한, 다양한 실험을 수행했음에도 불구하고 대부분의 결과가 비슷한 성능을 나타내어, 어떤 접근 방식을 선택해야 할지에 대한 판단이 어려웠다. 이로 인해 성능 향상에 대한 확신이 부족해 새로운 가설을 세우고 실험을 설계할 때 주저함이 있었고, 실험 과정이 지연되기도 했다.

-한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

• 구체적인 pipeline의 필요성

프로젝트의 초기 단계에서 전체적인 흐름을 체계적으로 관리할 수 있는 구체적인 파이프라인을 구축하지 못한 점이 여러 비효율을 초래했다. 이로 인해 모델이나 하이퍼파라미터를 선정하는 과정에서 많은 지연이 발생했으며, 이러한 지연은 프로젝트 일정에 부정적인 영향을 미치게 되었다. 앞으로의 프로젝트에서는 데이터 전처리, 모델 학습, 평가 및 테스트와 같은 각 단계별로 명확한 역할 분담과 진행 순서를 정해야 함을 느꼈다.

• 협업 도구로서의 깃의 활용

깃을 단순히 코드 저장소로만 사용하는 데에 그쳤던 과거의 방식에서 벗어나, 깃을 보다 적극적으로 협업 도구로 활용할 필요성을 느꼈다. 앞으로는 깃의 브랜치 전략, 이슈 관리, 그리고 코드 리뷰 등의 기능을 더 철저히 익히고, 팀원들과 깃 사용 규칙을 사전에 합의하여 협업 프로세스를 체계적으로 정립하려고 한다.

• 체계적인 서버활용

팀은 총 6명이지만, 주어진 서버는 단 4개에 불과하여 서버를 활용하는 데 있어 약간의 어려움이 있었다. 팀원 각자가 필요로 하는 자원을 적시에 확보하는 것이 쉽지 않았고, 이는 모델 학습 및 실험 진행에 영향을 미치는 요소가 되었다 또한, 각 서버의 이름이 임의로 정해져 있어 서버를 구분하는 데 어려움이 있었다. 각 서버별로의 역할을 명확히 정의하고, 서버 이름도 각 서버의 주요 기능이나 용도에 따라 지정하고, 구체적이고 직관적으로 지어 혼란을 줄여야겠다고 생각한다. 그리고 서버의 메모리가 오버되지 않는 선에서 적절히 사용될 수 있도록 배분해야겠다고 생각한다.

👉유희석

- 내 개인적인 학습 목표

: 첫 번째는 인공지능을 다른 조원들과 함께 만들어가는 경험이 없었다 보니까 최대한 많은 시행착오를 겪으면서 이런 프로젝트들을 진행하기 위해서 무엇이 필요한지, 무엇이 중요한지 알아보는 것이었습니다.

두 번째는 이론으로 배웠던 내용들을 천천히 하나씩 적용해보는 것이었습니다.

- 내 학습 목표를 달성하기 위해 한일

1. EDA

가장 먼저 했던 것은 그 500개의 클래스 이미지가 무엇인지부터 확인했습니다. 어쩌면 미려한 방법일 수 있겠지만 500개의 클래스를 전부 확인하며 그 클래스의 특징과 클래스를 더 큰 그룹으로 묶어서 볼 수 있을지를 확인했습니다. 그것을 시트에 정리해서 팀원들과 공유했습니다

2. 데이터 전처리

강의에서 기본적으로 배웠던 **rotation**, **flip**, **blur** 등의 전처리들을 적용해보며 성능의 변화를 체크보았습니다.

3. Augmentation 적용

데이터 증강을 통해 강건한 모델을 만들어야 하지만, 가장 효과적인 증강기법들을 찾아야하는데, 이 때 다양한 증강을 간편하게 실험할 수 있는 코드를 만들었고, 다양한 라이브러리를 이용해서 조금 더 고도화된 증강 기법인 **mixup**, **cutout**, **cutmix**을 적용시켜 실험해보았습니다.

4. Modeling

EDA에서 동물관련 클래스가 약 40퍼센트 차지하고 있음을 알게 되었고, 분류 문제를 더 작은 분류 문제로 나누어 학습해보면 어떨까 라는 생각에 동물 클래스에서 가장 많은 비율을 차지하는 개 클래스들을 분류하는

모델을 따로 두어, 두개의 모델을 학습해서 결과를 내보는 시도를 해보았습니다.

- 학습 목표를 달성하기 위한 과정 중 깨달은 점

이론적으로 배운 것이 그대로 실제로 바로 적용되지 않을 것이라는 것은 알고 있었으나, 그것을 피부로 처음 느껴보았습니다. 하나의 문제를 분석할 때 예상되는 원인이 너무 많이 있으며, 그것이 정말 그 이유인지 확인하는 등의 기술이 너무 부족하다는 것을 알게 되었습니다. 또한 문서화의 중요성을 알고 있었으나 더 철저한 문서화의 필요성을 알게 되었고, 각 실험의 결과 **loss curve**, 모델 **pt**파일등의 관리를 더 철저히 해야겠다고 생각했다.

- 아쉬웠던 점

1. 계획의 부재와 방향성의 부재

자원들 모두 관련 경험이 없다보니 체계를 세운 뒤 진행하지 못해 초반에 각자 플레이를 하는 듯 진행되어 아쉬웠습니다.

2. 문서화

나름의 문서화를 하기 위해서 실험 파라미터들을 시트에 기록하며 진행했지만, 세부적인 실험 결과나 정보들을 관리하지 못했고, 과거의 기록을 다시 찾아보거나 우리의 가설, 주장의 근거로 삼기 위한 근거 자료들을 막상 찾아보려고 하니 마땅히 쓸 것이 많지 않아서 개인적으로 후회했습니다.

3. EDA에서 가설을 세우고 이를 확인하는 과정이 많지 않았음

초반에 이를 해보고자 했으나, 조금만 더 수정해보면 어떨까라는 생각에 빠져 실험을 이어가다보니 그 방향성을 잃게 되면서 EDA를 하면서 얻은 아이디어들을 많이 활용해보지 못한 격이 되어 버려 매우 아쉬웠습니다. 이렇게 하면 안된다는 것을 시작도 전에 알고 있었음에도 그러지 못한 제 모습이 아쉬웠습니다.

- 다음 프로젝트에서 해보고 싶은 것들

1. 초반에 편한 실험을 위해서 조금 더 모듈화된 코드로 리팩토링을 먼저 할 것입니다

2. wandb나 tensorboard를 이용해서 실험 기록을 꼼꼼히 해볼 것입니다.

3. EDA에서 가설을 세우고 그 결과를 분석하는 그 흐름을 체계적으로 진행해 문서화까지 하는 것을 다시 해볼 것입니다.