

Sketch 이미지 데이터 분류 랩업 리포트

CV17조

(김민환, 김준원, 민창기, 신석준, 이준학, 전인석)

1. 프로젝트 Wrap Up

1-1. 프로젝트 개요

Sketch 이미지 분류 경진대회는 주어진 데이터를 활용하여 모델을 제작하고 어떤 객체를 나타내는지 분류하는 대회입니다.

Computer Vision에서는 다양한 형태의 이미지 데이터가 활용되고 있습니다. 이 중, 비정형 데이터의 정확한 인식과 분류는 여전히 해결해야 할 주요 과제로 자리잡고 있습니다. 특히 사진과 같은 일반 이미지 데이터에 기반하여 발전을 이루어나아가고 있습니다.

하지만 일상의 사진과 다르게 스케치는 인간의 상상력과 개념 이해를 반영하는 추상적이고 단순화된 형태의 이미지입니다. 이러한 스케치 데이터는 색상, 질감, 세부적인 형태가 비교적 결여되어 있으며, 대신에 기본적인 형태와 구조에 초점을 맞춥니다. 이는 스케치가 실제 객체의 본질적 특징을 간결하게 표현하는데에 중점을 두고 있다는 점을 보여줍니다.

이러한 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 높이는데에 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.

● 프로젝트 주제

Sketch 이미지 데이터 분류

● 프로젝트 구현 내용, 컨셉, 교육 내용과의 관련성 등

500개의 Class로 이미지를 Classification하는 정확도를 높이기 위하여 데이터 전처리, 데이터 증강, Loss함수, 모델훈련, 예측을 주된 목표로 삼고, 구현을 진행했습니다.

정확도만이 평가항목이므로, 예측 및 학습 속도를 무시하고 최대한 정확한 모델 사용과 앙상블에 방점을 두고 개발했습니다.

교육 내용 중 이미지에 대한 이해, 이미지 전처리, ConvNext 및 다양한 모델 선정 기준부터, 효과적인 학습 방법과 평가 방법등의 다양한 기술을 활용하여 정확도를 높이는 시도를 했습니다.

- 활용 장비 및 재료(개발 환경, 협업 툴 등)

팀 구성 및 컴퓨터 환경 : 6인 1팀, 팀당 총 4개의 V100 서버를 VSCode와 SSH로 연결하여 사용.

협업 환경 : 구글 스프레드 시트(각자 사용한 방식, 하이퍼파라미터, 정확도 등 공유)

의사 소통 : Zoom, 카카오톡

- 프로젝트 구조 및 사용 데이터셋의 구조도(연관도)

프로젝트 구조:

데이터 전처리 -> 데이터 증강 -> 모델 훈련 -> 모델 평가 및 검증 -> 하이퍼파라미터 수정

사용 데이터셋 구조:

- 훈련 데이터셋 : 한 클래스 당 29~31 장의 이미지로 구성되어 있으며, 500개의 클래스를 사용합니다. 총 15,021장으로 구성되어 있습니다. 이미지 크기 및 품질을 일관성 있게 유지하기 위해 전처리 및 증강 작업이 수행됩니다.
- 테스트 데이터셋 : 총 10,014장의 이미지로 구성되어 있습니다. 이 데이터셋은 모델의 일반화 성능을 평가하기 위해 사용되며, 훈련 과정에서 모델이 보지 못한 새로운 이미지로만 구성됩니다.

훈련 및 테스트 데이터셋 모두 전처리 단계에서 크기 조정 및 노이즈 제거 작업이 이루어졌으며, 데이터 증강을 통해 다양한 변형(회전, 확대, 밝기 조절 등)을 적용하여 모델의 견고성을 높였습니다.

훈련 데이터는 StratifiedKFold를 통해 여러 폴드로 나누어 교차 검증이 이루어지며, 각 폴드마다 데이터의 클래스 비율을 유지하여 훈련 및 검증 데이터로 사용됩니다.

모델 훈련이 끝난 후 테스트 데이터셋을 활용하여 모델의 최종 성능을 평가하고 결과를 분석합니다.

1-2. 프로젝트 수행 절차 및 방법

마일스톤

1. 강의 수강 및 기초 시도 (9월 10일 ~ 16일)

- 목표: 스케치 데이터 분류 대회와 관련된 기초 지식 습득 및 각자 기본적인 시도 진행.
- 세부사항:
 - 제공된 강의 수강
 - 과제 수행 및 이론 학습
 - 각자 선택한 모델이나 방법론으로 기초적인 실험 진행 (데이터 전처리, 간단한 모델 구현 등)
 - 각자의 시도 공유 및 피드백 세션 준비

2. 데이터 분석 및 전처리 (9월 17일 ~ 18일)

- 목표: 제공된 스케치 데이터를 분석하고, 전처리 과정에서 문제를 해결.
- 세부사항:
 - 데이터 탐색(EDA)
 - 데이터 증강 방법 탐색
 - 기본적인 전처리 코드 작성 및 적용

3. 모델 선정 및 기초 모델 개발 (9월 19일 ~ 21일)

- 목표: 각자 시도한 내용을 바탕으로 협의 후 최적의 모델을 선정.
- 세부사항:
 - CNN 계열, Transformer 계열 등 딥러닝 모델 중 적합한 모델 선정
 - 학습 데이터 준비 및 기초 모델 트레이닝
 - 모델 성능 평가 및 기초적인 결과 도출

4. 모델 최적화 및 개선 (9월 22일 ~ 24일)


- 목표: 모델 성능을 개선하기 위한 하이퍼파라미터 튜닝 및 최적화.
- 세부사항:
 - 최적의 하이퍼파라미터 탐색
 - 데이터 증강 및 과적합 방지 기법 적용
 - 모델 성능 재평가 및 성능 개선

5. 기존 모델 앙상블 (9월 25일 ~ 26일)

- 목표: 여러 모델을 앙상블하여 최종 결과물 도출.
- 세부사항:
 - 기존에 실험한 모델들을 앙상블 기법 적용 (Bagging, Boosting, Voting 등)
 - 최종 모델 평가 및 성능 검증

1-3. 프로젝트 수행 결과

<public 리더보드>

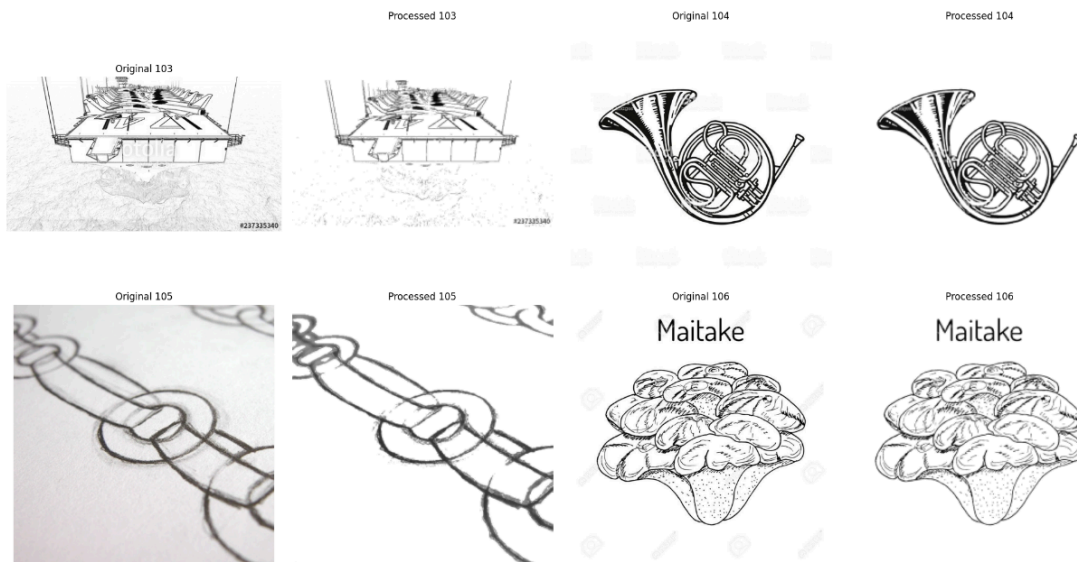
순위	팀이름	팀멤버	Accuracy
내등수 9	CV_17조		0.9310

<private 리더보드>

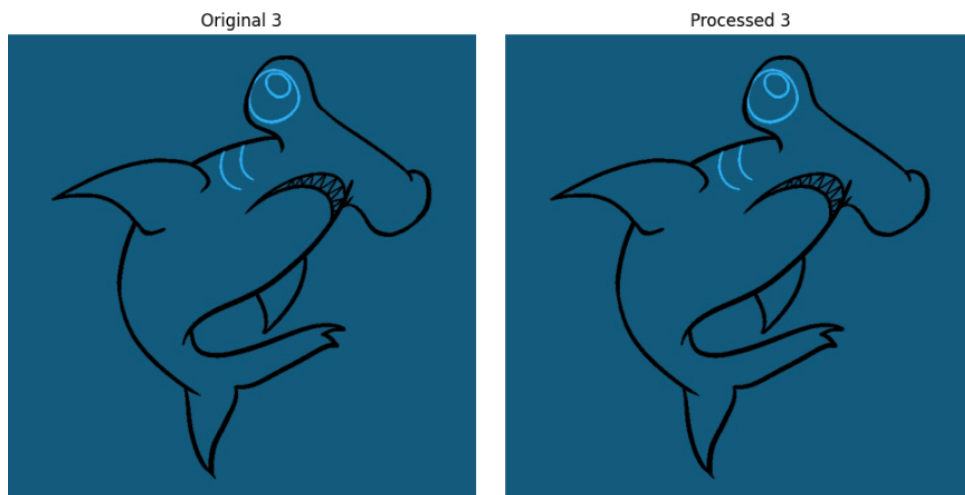
◦ 탐색적 분석 및 전처리(학습 데이터 소개)

<픽셀값을 이용한 데이터 전처리 결과>

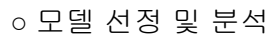
흑백 데이터로 판단(각 픽셀들이 채널별 값 차이가 크지 않은 경우) 되는 경우 3채널의 RGB값이 의미가 없다고 판단하여, **GrayScale**로 변환 했습니다. 노이즈(열은 워터마크와 배경)를 제거하기 위해 픽셀값을 이용하여 노이즈를 제거했습니다. 그 후, **N*N**으로 이미지를 맞추기 위하여 종횡비가 깨지지 않게 **Resizing** 및 **Padding**을 진행했습니다. (진한 워터마크에 대비하기 위해서 증강 때 진한 투명도를 가진 랜덤텍스트를 이미지에 생성하였습니다.)



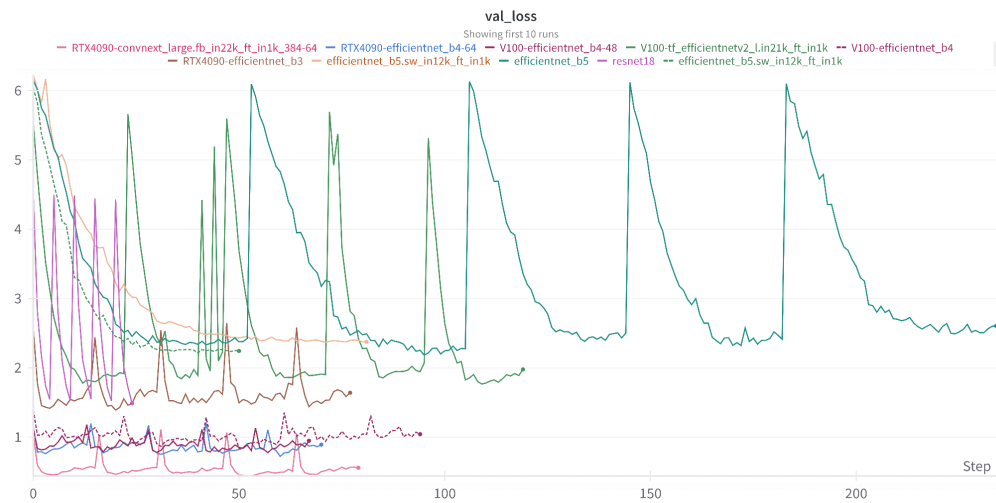
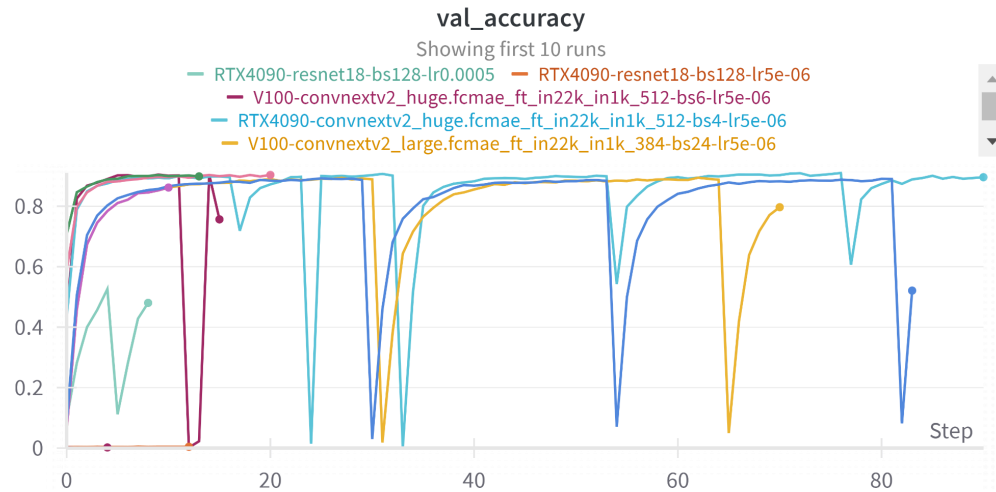
컬러 이미지로 판단(각 픽셀들이 채널별 값 차이가 큰 경우)되는 경우 컬러의 정보가 중요하다고 판단하여 **Resizing** 및 **Padding**만 진행했습니다.



모델의 크기가 커질수록 쉬운 이미지에 대해 과적합이 발생하는 것을 관찰하고, 이를 해결하기 위해 워터마크를 추가하고 다양한 크롭, 회전, 아핀 변환 등을 적용하여 모델 성능을 향상시키고자 했습니다.



- **Papers With Code**의 Classification 모델 성능 비교표를 참고하여 **CNN**모델 중 성능이 좋은 모델을 선정하였습니다.
(<https://paperswithcode.com/task/image-classification>)
- 500개의 다양한 클래스로 분류를 진행하는 프로젝트이니 만큼 크고 무거운 모델을 사용해야겠다고 판단하였습니다.
- **EfficientNet**과 **ResNet**, **ConvNext** 등 CNN 계열의 성능이 잘 나와 이 모델들에 집중해 여러 버전들로 성능 테스트를 진행했습니다.
- 이후 가장 성능이 잘 나온 모델들인 **ConvNext Base**, **Resnet 152**, **EfficientNet B4**, **ConvNext Huge**, **Inceptionv3**로 앙상블을 진행해 최종 결과를 냈습니다.
- **Validation Accuracy** 기준 단일 모델로 가장 성능이 높았던 모델은 **Convnextv2 huge**였습니다.



<모델 성능>

모델	정 확도
Resnet50	0.6930
Resnet101	0.8290
Resnet152	0.8421
Efficientnet_b0	0.7120
Efficientnet_b3	0.8240
Efficientnet_b4	0.8523
Efficientnet_v2_rw_m	0.8260

Efficientnet_v2_rw_t	0.8020
Inceptionv3	0.8234
Convnextv2_base	0.8810
Convnextv2_large	0.9160
Convnextv2_huge	0.9260

1-4. 자체 평가 의견

- **잘한 점들**
 - 팀원들이 프로젝트 각 단계에 적극적으로 참여하고 끝까지 완수하여 실전 지식과 경험을 쌓았습니다.
 - 각 팀원이 데이터 전처리, 모델 구축, 하이퍼파라미터 튜닝, 결과 분석 등을 수행하며 전반적인 프로젝트 흐름과 **competition**의 진행 과정을 이해했습니다.
 - 협업을 통해 **ImageNet-sketch competition**에 참여하고 문제 해결 능력을 키웠습니다.
 - 마지막에 시간이 부족했음에도 팀원들이 집중력을 발휘하며 협업하여 모델 성능을 많이 끌어올려 좋은 성과를 냈습니다.
 - **Google sheet**를 활용하여 학습 진행 상황을 관리, 공유했습니다.
 - 학습 결과 성능에 상관없이 가설을 세우고 결과를 내어 이론과 실전의 차이를 피부로 느꼈습니다.
- **시도했으나 잘되지 않았던 것들**
 - **Vision Transformer**를 활용한 모델들의 학습이 잘 진행되지 않았습니다. **Learning Rate**를 조정하거나 학습을 위한 추가적인 기법들을 공부하며 적용해보아야 할 것 같습니다.
 - **Git** 활용.
아직 **Git** 활용이 익숙하지 않아서, 이번 프로젝트 때는 적극적으로 활용하지 못했습니다. 다음 프로젝트 때는 **Git**을 활용하여 효율적인 협업이 될 수 있도록 해야겠습니다.

- 하이퍼 파라미터 최적화
Random Search, Bayesian Optimization, Hyperopt, Optuna 등 여러 최적화 방법이 있었지만 Grid Search 만을 사용했습니다. 다음에는 여러가지를 써보도록 해야겠습니다.

- 아쉬웠던 점들
 - 베이스라인 코드나 팀원들끼리 모델의 예측값을 공유하는 방법 등 팀 컨벤션을 명확히 정하지 못했습니다.
 - 특정 모델의 성능이 잘 나온다면 왜 잘 나오는 것인지, 앙상블 시에 사용한 모델들이 왜 좋은 시너지를 내는지에 대한 분석이 부족했습니다.
 - 앙상블을 미처 적용하지 못한 모델들이 있어, **accuracy**를 조금 더 향상시킬 수 있음에도 불구하고, 리더보드에 제출하지 못한 점이 아쉬웠습니다.

- 프로젝트를 통해 배운 점 또는 시사점
 - 다양한 모델을 실험하면서 성능 차이를 확인하고, 성능 비교 분석 경험을 쌓았습니다.
 - **Detection** 모델의 발전 과정을 통해 모델의 구조를 알고 데이터 특성에 따른 객체 감지 성능과 효율성을 높이는 방법을 배웠습니다.
 - **Zero Convolution**을 활용하여 작은 영역에서의 연산 효율을 높이는 방법을 배웠습니다.
 - **Positional Encoding sin, cos** 함수로 위치 정보를 인코딩하는 방식을 통해 입력 데이터의 순서를 인식하는 능력을 길렀습니다.
 - **Mix Precision**방법을 사용하여 GPU메모리 효율을 최적화하고 학습 속도를 개선하는 경험을 쌓았습니다.
 - **StepLR**을 통해 학습 후반부에서 모델의 안정성을 유지하는 방법을 배웠습니다.
 - **Label Smoothing**을 통해 모델의 일반화 성능을 향상시키는 능력을 길렀습니다.
 - **Gray scale**, 워터마크 추가, 워터마크 제거 등 다양한 이미지 전처리 기법을 적용하여 데이터 다양성을 확보하고 성능을 개선하는 방법을 배웠습니다.

- SSH서버의 **bash**명령어를 활용한 프로젝트 관리와 **vscode**를 이용한 SSH서버 활용을 통해 효율적인 시스템 운영 경험을 쌓았습니다.
- 앙상블시 여러 모델을 조합해 성능을 비교하여 다양한 아키텍처의 모델을 사용하는 것이 더 효과적이라는 것을 배웠고 코드를 돌려보며 앙상블의 개념을 직접 적용하고, 정확도 높은 모델이 항상 최선이 아님을 배웠습니다.
- Rotate, Affine, ElasticTransform, Erosion, Dilation, GaussNoise, MotionBlur, CoarseDropout, CLAHE, ShiftScaleRotate, RandomBrightnessContrast와 같은 다양한 **augmentation**기법을 통해 성능을 향상시키는 방법을 학습했습니다.
- 다양한 학습 스케줄러를 적용해 성능 조정 능력을 길렀습니다.
- 하이퍼파라미터 튜닝을 통해 모델 성능을 최적화하는 경험을 쌓았습니다.
- **Fine-grained Classification**을 공부하고 세밀한 객체 분류에 대한 이해를 깊이 하여 더 정교한 모델 구축 능력을 길렀습니다.
- 다음 프로젝트의 목표
 - Wandb 사용하여 모델 성능 공유하며 협업하기
 - 모델, 하이퍼 파라미터, 데이터 증강방법 등 범위를 정해 놓고 그 안에서 체계적으로 실험하기
 - **early layer freezing** 후 학습 시도해보기
 - 앙상블을 효율적으로 하기 위해 훈련 **output**은 **numpy** 파일로 공유하기
 - 프로젝트 시작 시 파이썬 모듈화 진행해서 작업환경 통일하기
 - Github에 **main code** 올려놓고 **branch** 만들어서 실험 작성하기 (자체 베이스 라인 코드 만들어 놓고 시작)
 - 앙상블 시 **soft voting**, **hard voting** 모두 고려해보기(k fold 시에도)
 - Notion에 서버 사용 기록 남겨서 체계적으로 사용하여 GPU메모리 부족 해결하기
 - 팀원 전부가 Git을 복습해서 Git을 협업툴로 적극 활용하기

2. 개인 회고

민창기

나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?

내 학습 목표는 전반적인 AI 개발의 흐름을 파악하고, 정확도를 높이기 위한 여러가지 방안들을 시도해 보는 것이었다.

강의들을 통해 이론적인 지식이 있는 상태에서 잘 짜여진 **BaseCode**를 받아 분석했다. 추가적으로 필요한 부분들은 실험을 하며 구글링 하였다.

데이터전처리와 테스트 방법을 위주로 고민 및 개발하였다.

나는 어떤 방식으로 모델을 개선했는가?

1. 데이터 전처리

데이터셋과 및 분석하며 문제정의를 하였다. 다음과 같은 조건과 의미를 지닌다.

- 주어진 이미지는 3채널이지만 무채색의 스케치 데이터를 기반으로 한다.
 - 흑백 변환을 하였다.(변환 후 배경이 흑색이고 스케치가 흰색인 데이터가 있고, 그 반대인 이미지도 존재한다. 이미지픽셀값을 기반으로 통일성을 맞추었다.)
 - 무채색 스케치 뿐 아니라 색이 의미 있는 컬러 이미지도 존재한다. 컬러이미지로 판단(각 픽셀들이 채널별 값 차이가 크지 않은 경우)되면 흑백변환을 진행하지 않는다.
- 각 클래스는 29~31장의 적은 데이터를 가진다.
 - 원본데이터와 증강데이터를 합쳐 2배의 데이터를 훈련시켰다.
- 이미지에 워터마크, 텍스트, 배경색 등 다양한 노이즈들이 존재한다.
 - 열은 워터마크와 배경을 RGB값을 기준으로 모두 흰색으로 바꿔 없앤다.
 - 진한 워터마크는 전처리에 진한 텍스트를 이미지에 추가하는 방식으로 보완한다.
- 이미지의 사이즈 및 종횡비가 제각각이다. (강제로 스케일링하면 종횡비가 갖는 의미가 깨질 수 있다.)
 - 이미지의 종횡비를 맞추는 최대선까지 **Scaling**을 진행하고 앞서 배경을 흰색으로 만들었으니 $N*N$ 을 맞추는 나머지 부분은 흰색 패딩을 채운다.

모델선택:

- 500개의 꽤 방대한 클래스를 가진다. 무거운 모델을 사용하는 것이 효과적일 것이다.
 - **ConvNext_Large** 모델 사용.

추론:

- 평가기준은 **Accuacy**뿐이다. 정확도만 높이면 되는 문제이므로 훈련, 추론 시간이 얼마나 걸리든 상관이 없다.
 - 앞서 훈련 데이터셋 구성 시 증강데이터와 원본데이터를 합쳐서 데이터셋을 구성하였다. 추론 시에도 증강데이터와 원본데이터를 각각 추론하고 두 결과를 앙상블한다. 그 이후 **K-Fold**를 이용해 훈련된 모델들을 다시 앙상블한다. 앙상블 방식은 두 과정 모두 확률이 높게 나온 결과에 가중치를 두는 방식을 사용하였다. (시간이 부족해 결과를 제출해보진 못했습니다.) => 이 방식이 **GIT**에 코드로 올라가진 않았으나, 유효한 방식인지 궁금합니다.

하이퍼 파라미터튜닝 및 스케줄러 선택:

- **Learning Rate, Label Smoothing, Weight_Dacay**를 위주로 하이퍼파라미터튜닝을 진행하였다. 첫 에폭의 **Loss**를 확인하고, 훈련이 진행될 수 있는 최대 **Learning Rate**를 시작점으로 잡았다.(**Loss**가 너무 높으면 훈련이 진행되지 않음.)
- 스케줄러: 다양한 스케줄러를 적용해보았고, 최종적으로 **cosine annealing warmup** 방식을 적용하였다.

내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?

데이터의 학습량을 2배로 늘리고, 데이터의 노이즈를 제거하였으며, 추론 시에도 다양한 앙상블을 시도하고자 했다.

데이터 전처리하는 부분은 실험을 했지만, 학습량을 늘리고, 추론 시 증강 데이터와 원본데이터를 앙상블하는 시도는 구현만 하고 시간분배 실수로 인하여 실험을 하지 못하여 깃허브에 올리지 못했다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

팀워크를 활용을 잘 못했다. 시간배분을 잘 못해 원하는 결과를 제출해보지 못했다. 모델 선택 시 디테일한 이유로 모델을 선택하지 못하였다. 처음에 **BaseLine**을 선정하는데 오래걸려 실험이 중구난방이 되었다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

팀워크를 활용. 시간 계획 짜기. 모델 선택 시 다양한 이유 탐색. 선택한 모델로 **BaseLine** 구축을 먼저 해야한다.