개인 회고_이현진

학습 목표 및 과정 + 모델 개선 방안

학습 목표

- 협업 프로젝트 경험이 처음인 사람부터 다양한 사람들이 모여있던 팀으로, `프로젝트`에 있어서 익숙해질 수 있다.
- 다양한 `SOTA 모델` 및 `loss function`, `optimizer`에 대해 알아 볼 수 있다.
- 프로젝트를 하기 위한 `Baseline 이해도`를 높일 수 있다.
- 다양한 Data Set에 대해 과적합 없이 `정확도`를 높이는 아이디어를 생각해 볼 수 있다.

학습 과정 및 사용한 지식과 기술

- Special Mission에 따라 EDA, Dataset/Data Generation, Model, Traning/Inference, Ensemble까지 순차적으로 코드를 작성해 나갔다.
- 데이터 EDA 결과 age에 대한 불균형이 심해, 이미지 데이터의 경우 imbalance의 해결 방법들을 탐색해보았다. **4 Ways to Improve Class Imbalance for Image Data**
  - Chipping instead of downsampling
  - Merging near-identical classes
  - Resampling specific classes
  - Adjusting the loss function

  제공된 Data Set에 대해서는 oversampling 및 imbalance data에 적합한 focal loss를 사용해보았다. 또한 age boundary를 변경해보았다.
- Augmentation(CLAHE, oversmapling, cutmix, facecrop)의 최적의 조합을 찾아냈다.
- 다양한 모델들의 결과를 hard voting 및 soft voting을 사용하여 결과를 도출해냈다.

시도한 점

- 다양한 Model에 다양한 optimizer, loss를 적용하여 비교하기 어려웠던 상황을 `Wandb`의 team project를 넣어 실행하였다. 같은 Model에 대해서 하나의 그래프에 Loss, F1 score를 한 눈에 확인할 수 있어서 편리하게 분석이 가능하였다.

- 하나의 Dropout이 아닌 여러개의 Dropout+Fully Connected+(Softmax+Loss Function)을 병렬적으로 진행하는 `Multi Sample Dropout`를 시도해보았다. 이는 하나의 batch에 대해서 다양한 결과값을 낼 수 있고 이를 평균을 통해서 최종 결과가 나오는 방식으로 자체적인 Ensemble을 한다고 생각할 수 있다. 실험을 해보기 전, Ensemble을 통해서는 항상 결과가 더 잘 나올 것이라 생각하였지만 다양한 Dropout rate 값에 대해서 실험해본 결과 Multi Sample Dropout을 사용하지 않은 Baseline에서의 성능이 더 좋게 나왔다.
- 더 다양한 SOTA 모델 및 optimizer를 사용해보기 위해서 `timm` 라이브러리를 사용하였다. timm 라이브러리의 모델들은 `timm.create_models()`의 num_classes라는 hyper parameter를 이용하여 자신의 목적에 맞는 output을 생성해 낼 수 있다는 장점을 가지고 있다. 이를 통해 더 성능이 높은 model들을 실험해 볼 수 있었고 발견할 수 있었다.

배운 점

- Jupyter notebook을 탈출하여 하나의 `Python Project`의 구조 및 실행 방법에 대해서 배울 수 있었다.
- f1은 score를 측정할 때만 사용할 수 있을 것이라 생각했는데, 1-f1 score를 통해서 f1 loss를 사용할 수 있다는 사실을 알 수 있었다.
- Terminal에서 학습 과정 및 추론 과정을 실행시킬 때, `argparse` 라이브러리를 사용하여 코드 상에서 명시적으로 집어넣는 것이 아닌 인자를 생성하여 편리하게 변수들을 다양하게 실험해볼 수 있다는 것을 알 수 있었다.

```
import argparse
parser.add_argument('--optimizer', type=str, default='Adam', help='optimizer type')
```

- 실험 결과가 좋지 않게 나오면 단지 해당 기법을 사용하지 말아야겠다는 생각만 가지고 있었다. 하지만 상위권 팀들의 발표를 듣고난 후, 성능이 향상되지 않은 이유 및 이를 대체할 만한 방안은 없는지에 대한 고찰이 필요하다는 것을 배웠다.
- f1 score가 좋은 모델들에 대해서 `Ensemble`을 진행하였다. public score에서는 향상이 있었지만, private score에 대해서는 성능의 차이가 컸다. 이는 너무 많은 Ensemble에 의한 Overfitting이라고 예상되어진다. Ensemble 뿐만 아니라 아무리 좋다고 하는 기능일지라도 과도하게 적용하게 되면 Overfitting, 그리고 다양한 역기능을 초래하게 된다는 것을 알 수 있었다.

아쉬운 점+앞으로의 방향

- `Baseline Code`를 작성하는 데에 있어서 미숙했고 깔끔한 코드를 짜지 못했다. kaggle이나 dacon의 다양한 분들의 baseline code에 대해서 분석하고 공부하여 스스로 baseline code를 짤 수 있는 사람이 되어야겠다.
- 사용하는 `Model`에 대해서 완벽하게 파악하지 못한 점이 아쉽다. 다음번에는 논문과 함께 어떤 점에서 더 효율적인 성능 향상이 있고, 해당 Model을 스스로 Custom 해보는 능력을 길러야겠다.
- 최적의 optimizer, loss function, hyper parameter를 선정할 때, 하나씩 바꿔가면서 실험을 통한 F1 score 값을 확인해보며 찾아갔다. 앞으로 있을 프로젝트에서는 Wandb의 `sweep` 및 `Ray Tune`을 사용하여 효율적인 `hyper parameter tuning` 실험을 진행해보아야겠다.
- 성능을 올리는 데에 있어서 적절한 `Augmentation`을 사용하는 것도 중요하다. 따라서 다양한 Augmentation 기법들을 적용해보고 실험을 해서 최적의 Augmentation 조합 및 순서를 알아내야한다. 하지만 그 만큼 시간이 오래 걸린다. 이를 위해서 `Auto Albument`라는 AutoML tool이 존재한다. 이는 image classification과 semantic segmentation의 작업에 대해서 제공한다. 이번 image classification 프로젝트에서 적용해보지 못했지만 semantic segmentation 프로젝트에 적용해보아야겠다.
- 매일 진행한 실험 내용 및 결과, 그리고 배운 점들을 정리 및 기록하지 못한 점이 아쉽다. 다음 프로젝트에서는 `프로젝트 일별 보고서`를 위한 형식을 정해두고 매일 기록해보는 습관을 들여보아야겠다.