개인 회고 : 김소연

모델 개선과 학습 목표 달성을 위해서 시도해본 것, 그에 따른 결과들 :

☺ **클래스 불균형을 해결하기 위한 다양한 손실 함수 적용**

과거의 저는 클래스 불균형이 존재하는 데이터셋 해결에 관심이 있었고, 특히 손실 함수로 해결하는 방법이 fancy하다고 생각했습니다. 문제는 대부분의 논문들이 사용하는 데이터셋은 open benchmark 데이터셋에서 인위적으로 클래스 불균형을 주거나, 본 프로젝트 데이터셋보다 클래스 간 특징이 분명한 데이터셋입니다. 그래서 실제 현업에서 마주하는 데이터의 극단적인 불균형, 클래스 간의 모호성, noisy 함이 존재할 때도 적용될 수 있는가?에 대한 호기심이 있었습니다. 따라서, focal loss, F1 loss, label smoothing, LDAM loss, weighted cross entropy loss를 efficient net b0에 적용해보았습니다. F1 loss 는 눈에 띄게 성능이 좋지 않았고 다른 Loss들 역시 성능 향상의 경향을 파악하기 힘들었습니다. 그나마 높아 보이는Focal, LDAM loss 경우 리더보드에 제출했었으나 cross entropy보다 낮게 나왔습니다.

☺ **클래스 불균형 해결을 위한 weighted sampler 적용**

학문적 측면에서 논문은 손실 함수를 많이 만지지만, 현업에서는 데이터 자체의 균형을 맞춰주는 것이 중요하다라는 의견이 많았습니다. 따라서, 클래스 분포에 따라 weight를 계산하고, 각 클래스별 weight에 따라 다른 확률로 데이터가 샘플링 되도록 하기 위해 1) pytorch의 `weightedRandomSampler`, 2) MIT에서 제공하는 `ImbalancedDataSampler`를 수정해 사용했습니다. 그러나 적용하지 않은 것보다 f1 score가 낮게 나왔고, focal loss보다 낮은 성능을 보였습니다.

☺ **기존 모델을 수정하고, 다른 task 를 수행하는 다른 모델과 multi-task learning 으로 학습**

기존 efficient net model에서 dropout 을 강하게 주기 위해 추가 FC layer를 붙여서 실험해보았고, 성능적 향상이 있었습니다. 나이별 데이터 불균형이 심하고, 이 부분이 본 테스크의 중요한 부분이기 때문에 classification보다 더 높은 정보량을 주어 모델을 학습시킬 수 있는 linear regression 모델을 학습시키고자 했습니다. 단, model의 feature extractor는 공유하고, 한쪽 FC는 전체 클래스 분류를, 한쪽 FC는 age regression을 수행하는 multi-task learning으로 모델링 했습니다. 처음에는 age regression loss가 너무 압도적이었고 그 때문에 클래스 분류 성능이 굉장히 떨어져서(10%미만) age regression loss에 0.001을 곱해주었습니다. 학습이 다시 안정적으로 되긴 했지만, confusion matrix를 확인해보면 해당 regression 모델을 붙여주는게 오히려 60세 이상이 포함되는 클래스(2,5,8,11)에 대해 떨어지는 성능을 보여주었습니다.

☺ **EDA, pandas 를 통한 데이터셋 구성, Cross validation 과 앙상블을 위한 베이스라인 데이터셋 모듈 수정**

보통 benchmark 데이터셋이나 폴더로부터 클래스를 주는 방식으로만 학습해봤기 때문에 훈련 데이터 EDA, pandas로 Dataset 모듈 짜는 것이 처음이었는데, 무슨 일인지 모르겠지만 베이스라인 코드보다 제가 짠 코드로 했을 때 성능이 더 높게 나왔기 때문에 single model 학습 시 제가 짠 데이터 모듈을 계속 사용했습니다. 이후 앙상블, cross validation을 위해 `sklearn`의 `train_test_split` 메소드 사용 대신 기존 베이스라인 코드에서 'k' fold 만큼 각 train,val데이터셋을 반환하는 것으로 수정해서 적용했습니다. 학습은 정상적으로 돌아갔습니다.

☺ **앙상블과 TTA 코드 구현과 적용**

앙상블이 개념적으로는 쉽지만 코드로 한번도 구현해보지 않아서 어떻게 데이터를 쪼개고, prediction을 수합하는지 와닿지 않았습니다. Special mission 코드와 수정한 k-fold split 데이터셋 코드를 통해 제 학습 코드에 돌아갈 수 있도록 앙상블과 TTA 를 마지막날에 구현했습니다. 시간이 오래걸림에도 불구하고, 중간 디버깅을 제대로 하지 않아 실험 중간에 멈춰야하는 상황이 발생했습니다. 제출 임박에 돌아가던 코드의 학습 경향성이 좋았음에도 시간 내에 학습하지 못해 제출하지 못했습니다.

깨달은 점, 아쉬운 점 :

과연.. **위의 방법들이 정말 성능 향상을 주지 않은 것일까요?** 사실은 수행 과정에서 기본적인 부분이 빠져있었기 때문이라 생각합니다. 먼저 validation loss에 training loss를 텐서보드에 넣어서 loss 경향성을 완전 잘못 파악했습니다. Loss가 잘 줄었기에 초반 모델, 하이퍼파라미터를 정하는 단계의 실험에서는 validation accuracy, f1 score만을 기준으로 판단했고 그 다음 실험을 이어갔습니다. 이후 validation loss를 정확히 수정한 후 진행한 실험에서 loss가 금방 증가하는 overfitting을 관찰했고, 단지 방법론 측면에서 그것을 수정하려고 시도했습니다. 마지막 날 앙상블을 시도하면서 문득 learning rate를 더 줄여야 하는게 떠올라서 휴리스틱하게 줄여서 실험했으나 시간 내에 학습되지 못해 제출을 못하는 불상사가 발생했습니다.TT. 또한, 저는 데이터셋이 많은 것이 좋다 판단해 validation loss, accuracy 파악 후 해당 셋팅으로 전체 데이터셋에 학습시키고 비슷한 에폭에서 멈추는 접근으로 실험을 수행했습니다. 거의 눈감고 더듬듯 실험하는 모양이니 답답했고 도대체 다른 사람들은 어떻게 하는지 궁금했는데.. 팀 발표를 보니 거의 대부분의 팀이 training/validation으로 나누어 성능을 측정하고 그 성능에 근거하여 모델을 제출, 추가적으로 out of fold ensemble, hard voting을 수행했던걸 보고 이마를 쳤습니다. 사실 이 부분 때문에 팀원들이 어느 부분에서 학습을 멈춰야하는지 물어보는 것에 시원한 대답을 못줬는데, 제가 잘못된 방향으로 팀원들을 인도한 것 같은 마음이 들어 미안했습니다.

본 프로젝트를 통해 다음 프로젝트에서 시도해볼 것

이번 프로젝트에서는 진정한 의미의 협업보다 팀원 별 속도에 맞게 본인이 원하는 것을 다뤄봄으로써 이해도를 높이는 과정으로 보냈습니다. 따라서, 다음 프로젝트에서는 공동의 목표로 효율적인 분업으로 프로젝트를 수행해보고 싶습니다. 또한 lr_finder, reducedLROnPlateau메소드를 이용해 learning rate에 대한 판단을 정확히 한 후 실험을 넘어가려 합니다.