**T1058 문재훈 WrapUp 리포트**

본인의 점수 Accuracy : 77.3651%, f1score : 0.6718, 154등

● 검증(Validation)

1. 사람을 기준으로 Train Valid를 나누어보기도 하였고, 모든 이미지에 대하여 랜덤으로 나누기도 해보았습니다. 사람 기준으로 나누는 것이 더 좋을 것이라 예상하였지만 (Leakage 문제 해결) 오히려 LB 점수를 보면 사람 기준으로 train valid split을 하지 않는 것이 점수가 더 좋다는 것을 알 수 있었습니다.

2. Valid set의 비율은 0.05, 0.1, 0.15, 0.2 등으로 시도해보았습니다. (최종은 0.2로 선택)

● 사용한 모델 아키텍처 및 하이퍼 파라미터

먼저, 저는 하이퍼파라미터들의 튜닝을 시도함에 있어서 가장 간단한 모델아키텍쳐 베이스라인을 잡아두고 (Resnet 50, Adam옵티마이저를 기반으로 학습한 모델) 튜닝하고 싶은 하이퍼파라미터를 제외한 모든 요인을 통제하고 하나씩 바꿔가며 실험해보았습니다.

- 사실 이는 다양한 하이퍼파라미터의 상호작용으로 나올 수 있는 긍정적인 효과는 파악하지 못했기 때문에, 옳은 방법인지는 아직도 개인적으로는 의문이 듭니다.

1. 모델 아키텍쳐 : Resnet34, Resnet50, Resnet101, efficientnet\_b4 등 모델들을 사용하였습니다. 최종적으로는 efficientnet\_b4 모델을 채택하였습니다. (저의 실험결과로는 모델의 사이즈가 클수록 퍼포먼스가 더 좋아졌기 때문에 큰 모델을 가져와서 우리의 Task를 수행하는 것이 유리하다고 결론을 내릴 수 있었습니다, efficientnet\_b7 등 더 큰 모델을 학습시키지 못한 것이 아쉬웠습니다.)

2. ImageAugmentation의 경우에는 Train시와 Test시에 모두 380\*380 size로 CenterCrop만을 사용하였습니다. EfficientNet\_b4가 380\*380 사이즈로 학습이 이루어져있어서 CropSize를 이에 맞추는 것이 옳다고 생각하여 적용하였습니다. (Rotate, ColorJitter, RandomCrop, Resize 등 다양한 augmnetation을 시도해보았지만 CenterCrop을 제외하고는 성능이 좋지 않았습니다. 사실 이는 우리의 Task와 연관지어 생각해보면 당연한 결과라고 생각해볼 수 있었습니다)

3. Adam, AdamP, SGD, SGDP 등 다양한 옵티마이저를 시도해보았고 결국 Adam 옵티마이저를 사용한 모델을 채택하였습니다. (Adam보다 SGD가 성능이 더 좋다는 것은 실험을 하며 느낄 수 있었지만 SGD 옵티마이저를 기반으로 한 모델은 전체 학습 소요시간이 너무 오래 걸려 Adam으로 많은 실험을 해보고 최종결과를 낼 수 밖에 없었던 점이 굉장히 아쉬웠습니다.)

4. StepLR, CosinScheduler, CosinScheduler with WarmUp 등 다양한 스케쥴러를 시도해보았지만 스케쥴러를 적용하지 않은 모델이 오히려 성능이 더 좋았습니다. (학습시 검증결과와 LB결과 모두)

5. earlyStopping을 통해 valid에 대한 성능이 더 좋아지지 않으면 학습을 중지하는 식으로 과대적합은 피할 수 있게 설계하였습니다.

6. train batch size를 (16,32, 64, 128, 256) 다양하게 시도해보았고 최종적으로는 16으로 설정하였습니다. train batch size를 작게 하는 것이 모델성능을 끌어올리는데 중요한 역할을 하였습니다. (batch size를 크게 할수록 좋다는 일반적인 통념이 잘 적용되지 않은 이유로는 우리의 데이터셋 크기가 그리 크지 않았다는 점과 mini batch 학습의 장점을 통해 local minima를 벗어날 확률을 더욱 크게 만들어주는 것이 우리의 Task에서는 중요했을 것이라는 점을 꼽을 수 있을 것 같았습니다.)

7. learning rate는 1e-5로 최종 선택하였습니다. (1e-3, 3e-4, 1e-4, 3e-5, 1e-5 등) 많은 시도를 해보았는데 1e-5가 가장 성능이 좋았습니다. 학습율을 이보다 더 작게 두고 학습하는 실험은 해보지 못했다는 것이 아쉽지만, Scheduler를 통해 학습율을 줄여나가는 방식이 그리 잘 작동하지 않았다는 것을 보면, 어차피 base lr을 더 작게 하는 것은 모델 학습 시간을 과대증폭시키는 악영향만 미치게 되었지 않았을까 생각도 해보았습니다.

8. loss function을 cross entropy, f1 loss, focal loss, label smoothing 을 시도해봤었는데 당연히 f1 loss를 채택하는 것이 좋을 것이라는 예상과는 달리 cross entropy를 이용하는 것이 가장 모델 퍼포먼스에 좋은 영향을 주었습니다.

● 앙상블 방법

18개의 class로 생각하는 모델링이 아니라 3개의 Task로 각각 모델링하여 결과를 합산하여 도출한다거나 age에 대해서는 regression 문제로 접근한다거나, 단일 아키텍쳐 모델에 대하여 SWA를 시도해보거나, 혹은 여러 아키텍쳐의 모델들에 대해 앙상블 (단순평균 등)을 시도해보고 싶었지만 코드구현에 있어서 시간소요가 많아 완성시키지 못하여 아쉬움이 남았습니다. 대회는 종료되었지만 이번주말까지 이러한 기법들을 적용하는 코드 구현에 집중해보려고 합니다.

● 시도했으나 잘 되지 않았던 것들

1. minor class에 단순리샘플링을 통해 불균형 문제를 모델이 잘 처리할 수 있도록 설계해보았지만 오히려 성능은 떨어졌습니다.

2. train batch size를 크게 하는 것이 모델의 성능을 악화시켰습니다.(저는 16으로 최종 batch size를 채택하였습니다) train batch size가 더 작을수록 성능이 더 좋다는 의견이 피어세션 내내 제기되었습니다. (mini batch 학습의 특성상 local minima를 벗어날 수 있다는 장점이 존재하여 성능이 더 좋아질 수 있다고 알고 있습니다. 하지만 피어세션 중 나온 얘기로는 이번 Task는 터무니없이 작은 batch size(4와 같은)에서 성능향상을 이루어내신 캠퍼분도 계시다고 들어서 아무리 그래도 어떻게 이렇게까지 작은 batch size가 좋은 퍼포먼스를 낼 수 있었던 것인지는 개인적으로 잘 이해가 아직도 잘 되지 않습니다.

3. 저는 Scheduler를 추가한 모델 학습을 통한 성능향상은 무조건 이루어낼 수 있는 것이라고 했지만 제가 시도해본 StepLR, CosinScheduler, CosinScheduler with WarmUp 에서는 퍼포먼스의 향상을 이끌어내지 못했습니다. 아마 최대에폭수, min\_lr 등 다양한 하이퍼파라미터를 잘 조정하지 못한채 스케쥴러를 투입하여 성능향상을 이끌어내지 못한 것으로 생각을 하고 있습니다. 스케쥴러 관련 파라미터들을 바꿔가며 더 다양한 시도를 했다면 모델의 더 좋은 성능을 끌어낼 수 있지 않았을까 하는 아쉬움이 남습니다.

(그런데 혹시 딥러닝 모델 학습에서 어떤 스케쥴러를 적용하더라도 모델의 성능이 나아지지 않는 경우도 있긴 할까요? Local minima Problem이 없는 쉬운 모델학습의 경우가 바로 그러한 경우일까요?)

● 배울 수 있었던 점

베이스라인 코드를 이해하고 다양한 코드를 수정하고 추가해보며 vscode ssh 연결도 해볼 수 있었고 파이썬 프로젝트를 통해 터미널 창에서 간단하게 파이프라인을 돌려보며 파이썬 프로젝트의 강력함을 느낄 수 있었다는 점이 가장 저에게 도움이 되었던 점이라고 생각합니다. (한마디로 베이스라인코드를 보고 마스터님의 강의를 들으며 코드는 이렇게 짜야하는 것이구나를 느낄 수 있었습니다.)

이미지 데이터를 다루며 다양한 기법들을 적용하며 분석한 경험이 전무하였다 보니, 이번 Stage에서는 저의 실력이 아직 부족하여 하루하루 미션이 주어지는 것을 따라가기에 시간상 벅찼고 그로 인해 머리로 떠올린 다양한 기법들을 실제로 적용하는 시도를 해보지 못했다는 것이 굉장히 아쉬웠습니다. 이후 Stage에서는 더욱 성장한 모습으로 제 스스로 만족할 만큼 다양한 시도를 해보고 싶습니다.