Wrap-up Report

김종헌

이번 competition을 진행하면서 시도했던 여러 전략들, 그리고 최종적으로 활용한 방법에 대해 다루었습니다. 마지막에는 이번 대회를 진행하면서 느낀 점들을 담았습니다. 여러 학습 환경에 대해 기록을 제대로 해두지 않아 서술하는 방식으로 작성하였습니다.

최종 submission

Public LB	Accuracy **82.5556%**	F1 score **0.7861**	**2위**
Private LB	Accuracy **81.9048%**	F1 score **0.7729**	**2위**

시도

ver. 1	**Model** Backbone EfficientNet B0/EfficientNet B4/EfficientNet B7 Classifier FC layer (18 class) **Criterion – CrossEntropyLoss** **Optimizer – Adam** **Scheduler – None** **Augmentation – RandomCrop**

가장 간단하게 생각해볼 수 있는, 18 class를 한 번에 예측하는 모델입니다. 다만 성능이 다소 저조하였고 18 class를 한번에 예측하는 것은 각 클래스 예측에 예상치 못한 편향이 생길 수 있겠다는 생각이 들었습니다. 18 class라는게 (제 생각엔) 적지 않은 클래스 수였기 때문에 다음 버전부터는 한 번에 예측하기보다는 카테고리별로 나누어 예측을 할 수 있는 모델을 설계하였습니다.

한편 Image classification에서 EfficientNet의 버전이 높아질수록 성능이 좋아지는 것을 보고 더

무거운 모델을 적용해보았는데 학습 시간은 오래 걸리고 성능은 감소하는 결과를 얻었습니다. 이 점을 보고 데이터가 적은 상황에서 마냥 무거운 모델이 정답은 아니라는 점을 알게 되었고 무엇보다도 일단 다른 부분(데이터셋, augmentation 등)에서의 전략 결정을 위해 일단은 가벼운 모델인 EfficientNet B0를 계속해서 활용해야겠다고 생각했습니다.

첫 시도였고 v1을 설계할 때는 사실상 파이프라인을 어떻게 짤지에 대한 고민을 더 많이 했던 시간이었기 때문에 모델 설계 외(loss, optimizer 등)에는 가장 쉽게 적용할 수 있는 것들을 활용하였습니다. 이 과정에서 모델, 데이터셋, 학습, 추론 등 여러 코드들을 어떻게 짜야하는지 알 수 있었고, 필요한 기본적인 라이브러리(os, glob 등)들에 익숙해질 수 있었습니다.

ver. 2	**Model** Backbone EfficientNet B0 Classifier FC layer X3 (3 class/2 class/3 class) FC layer (18 class) **Criterion – CrossEntropyLoss** **Optimizer – Adam** **Scheduler – None** **Augmentation – CenterCrop, RandomRotate**

헤드 세 개를 두면 각 헤드에서 각 카테고리별 feature를 인식하지 않을까 하는 생각에 헤드 세 개를 두었고 마지막에는 이 8개 클래스를 통해 다시 최종 class를 예측하는 FC layer를 하나 더 두었습니다. 여기서는 그래도 성능이 앞선 v1보다는 잘 나왔지만 이 모델도 마지막에 결국 FC layer를 하나 더 통과하므로 각 헤드가 무엇을 분류해야 하는지 제대로 알지 못할 것 같았습니다. 즉, 결국 앞선 v1에서와 같이 예측에 의도치 않은 편향이 들어갈 것 같다는 생각이 들었습니다. 또한 순전히 '이렇게 하면 어느정도 괜찮지 않을까?'하는 근거 없는(?) 생각에 설계하게 된 모델이라 곧이어 폐기하게 되었습니다.

아직까지 그 외 요소들은 거의 수정하지 않았고, 다만 Augmentation에서 CenterCrop이 더 성능이 좋길래 360x200으로 centercrop하였고, RandomRotate도 -20~20으로 주고 학습을 진행하였습니다. 그런데 당시 RandomRotate가 성능에 긍정적 영향을 주는지는 확인하지 못했습니다. 그냥.. 좋아보이는거 썼습니다.

ver. 3	**Model** Backbone EfficientNet B0 Classifier FC layer X3 (3 class/2 class/3 class) if mask: return FC_mask(x) if gender: return FC_gender(x) if age: return FC_age(x) **Criterion – CrossEntropyLoss** **Optimizer – Adam** **Scheduler – None** **Augmentation – CenterCrop, RandomRotate**

v2에서와 다르게 마지막에 8개 클래스를 다시 18개로 늘려주는 FC layer를 두지 않은 모델입니다. backbone만 공유하고 **각 카테고리별로 학습을 따로따로 진행**하여 이를 최종 inference에 활용하였습니다. 그래서 모델 구조를 보면 어떤 카테고리를 예측하느냐에 따라 마지막에 활용하는 FC layer의 종류 및 output이 달라지며, 그 외의 FC layer들은 아무 동작을 하지 않습니다.

v3에서는 v2에 비해 성능이 개선되었지만 역시 문제점이 많습니다. 이렇게 백본만 공유하고 카테고리별로 학습을 따로 진행하게되면 multi-head를 둔 의미가 없어질 뿐더러 각 학습에서 역전파가 제각기 전달되므로 어떤 카테고리를 먼저 학습시키느냐에 따라 inference의 결과도 크게 달라질 수 있습니다. 사실 치명적인 문제지만 이 모델을 설계할 때는 아무 생각이 없었습니다.. ㅋㅋㅋㅋㅋㅋ

지금와서 핑계를 대보자면 아직 익숙하지 않아서 설계하게 된 모델입니다. 만약 하나의 input에 대하여 동시에 output 3개를 내놓는다면 loss는 설계를 어떻게 할 것이며 학습 코드 자체를 어떻게 짤지 좀 막막해서 이렇게 모델 설계를 하게 되었습니다. 하지만 이후 v5에서는 이러한 문제를 개선한 새로운 multi-head classifier를 만들게 되었습니다.

ver. 4	**Model** **(Multi model) Mask classifier, Gender classifier, Age classifier** Backbone EfficientNet B0 / ResNeXt50 Classifier FC layer (proper class(3/2/3))

Criterion – CrossEntropyLoss, Focal loss
Optimizer – Adam, AdamP, SGD
Scheduler – ReduceLROnPlateau, CosineAnnealingLR, StepLR
Augmentation –
CenterCrop, RandomBrightnessContrast, HorizontalFlip, Blur, RandomFog, ColorJitter, CLAHE, ElasticTransform, GridDistortion, ShiftScaleRotate

v4에서는 각 카테고리별로 모델을 분리하여 총 3개의 모델을 두었습니다. 같은 백본에서 세 클래스를 예측하는 것보다 아예 모델을 분리하는 게 더 좋은 전략일 것 같다는 생각이 들었고 실제로 모델이 mask와 gender는 잘 예측하는데 age 예측은 잘 못하고 있다는 것을 v4 모델을 활용하면서 깨닫게 되었습니다.

그래서 여기부터는 **age 예측을 어떻게 개선시킬지에 집중하였습니다**. mask, gender model은 성능이 잘 나와서 그냥 예측값까지 고정시켜두었습니다. age 예측을 위해 age filter도 적용하고 age model 학습시에 다양한 hyperparameter와 loss 함수, optimizer를 활용하였습니다.

age filter는 데이터 부족으로 인해 모델이 편향적인 학습을 할 것을 대비하여 라벨링의 기준을 변경하는 방법입니다. 토론 게시판 및 피어세션을 통해 이 방법을 알게 되었고, 결론적으로 58세 이상부터 노년층으로 간주하는 것이 가장 성능이 좋았습니다.

loss 함수의 경우 Focal loss를 추가적으로 활용해보았습니다. Focal loss의 경우 gamma 값이 조금만 높아도 초반 에폭에서 수렴이 안되길래 계속 2로 잡고 활용하였습니다. 그런데 성능 차이가 거의 없었어서 좀 당황했습니다. 뒤늦게 gamma를 5정도로 잡아도 수렴속도는 느릴지언정 성능은 더 잘 나올 수 있다는 이야기를 피어세션에서 들었습니다. ㅠㅠ CrossEntropyLoss에 weight을 주는 방법도 적용해보았지만 성능이 잘 안나왔고, 이렇게 하면 minor data에 overfitting이 일어날 수도 있다는 우려에 weight 적용도 하지 않기로 했습니다. 결국 결론적으로 가장 무난한 CrossEntropyLoss를 활용하기로 하였습니다.

optimizer의 경우 여러 고려 대상이 있었는데, 학습을 돌려본 결과 SGD 수렴속도가 확실히 느리긴 해서 시간이 한정되어있다는 점에서 일단 고려대상에서 제외하기로 하였고 토론게시판에서 본 AdamP를 활용해보았으나 별다른 차이가 없어 그냥 Adam을 활용하기로 하였습니다.

v4에서는 처음으로 **scheduler**를 도입하였습니다. 솔직히 스케줄러별 성능 차이는 체감하지 못했고 이 시점에서는 돌리는 모델이나 에폭을 계속해서 바꿔주면서 학습하였기 때문에 이런 요소들에 큰 영향을 받지 않을 수 있는 ReduceLROnPlateau를 활용하기로 했습니다. 수렴 속도에 따라 자동으로 decay 시점을 잡아준다는 점이 일단 초심자인 저에게는 매력적으로 다가왔습니다. lr을 얼마로 줄지조차 감을 못잡고 있었으니까요.. ㅠㅠ

augmentation도 v4부터 본격적으로 활용하였습니다. 먼저 albumentations 라이브러리를 활용하기로 하였고 어떤 기법을 사용하는 것이 좋을지 여러 실험을 해보았습니다. 처음에는 정말 많은 방식을 활용하였고, 이 과정에서 적힌 것들 중 RandomFog, ColorJitter, GridDistortion, Blur 등의 방법은 추후 폐기되었습니다. 먼저 GridDistortion이나 ElasticTransform과 같이 왜곡을 주는 것이 사람을 다르게 보이게 만들기 때문에 넣으면 좋겠다는 생각을 했는데 오히려 안좋았고, RandomFog, Blur, ColorJitter 등의 기법도 잘 먹히지 않았습니다. 지금 생각해보면 뭔가 피부톤이나 피부주름 등을 변화시킬 수 있는 요소들이 나이 예측에 부정적 영향을 준 것 같습니다.

또한 **소수 데이터만 증강시키는 방법도 활용**하였습니다. 일단 마스크 착용여부나 남녀 구분은 데이터 분포가 다소 왜곡되어 있어도 예측 성능이 좋았기 때문에 쟁점은 노년층 데이터만 증강시키는 것이었습니다. 이를 위해 dataframe에서 노년층 데이터만 리샘플링하였고 이렇게 리샘플링된 데이터에만 augmentation을 적용하였습니다. (original data에는 CenterCrop만 적용) 실제로 성능향상이 꽤 의미있는 수준으로 나왔고 사실 여기서 성능이 더 잘나왔기 때문에 Focal loss를 활용하는 방식이나 cross entropy에 weight을 주는 방식 등을 쉽게 포기할 수 있었습니다. 이렇게 소수 데이터만 증폭시키면 데이터의 분포가 왜곡되지 않기 때문에 이런 방법들이 필요 없었기 때문입니다.

아쉬웠던 점은 여기서 뭔가 의미있는 실험을 하지 못했다는 것입니다. 무언가를 변화시키면 통제변인을 명확하게 고정했어야 하는데, 한 번에 여러 요소들을 변경하였고 정확히 각 변화가 성능에 어떤 영향을 줬는지 확인하지 못했습니다. 그리고 이런 실험들을 진행하면서 기록을 명확하게 남기지 못했습니다. 그래서 v4를 개선시키면서 오랜 시간을 소모했지만 사용한 시간에 비해 얻은 성과가 별로 없었습니다.

사실 대회 마감 이틀 전까지도 이 모델만 고집하였습니다. 어느 정도 성능이 잘 나왔고 학습이 더 빠른 모델 구조는 있을지언정 이보다 예측을 잘하는 모델은 없을 것이다라는 생각이 들었습니다. 다만 초기에 설계한 모델에 비해 더 좋은 성능을 보이는 모델을 찾을 수가 없었고 일찍이 ensemble과 KFold에 눈을 돌렸습니다.

KFold는 5로 적용하였는데 솔직히 큰 성능 향상은 없었습니다. 다만 이 때 놓친 점이, KFold로 만들어진 5개의 모델을 ensemble했으면 좋았을텐데 저는 그냥 5개 중 F1 score 기준 best model만 활용했습니다. 당시는 5개 모델 ensemble에 대한 생각 자체를 좀 못했던 것 같습니다. 이런 잘못된 생각도 있었고, KFold는 학습시간이 5배로 걸리기 때문에 결국 마지막 제출까지도 KFold를 활용하지 않았는데 지금와서 좀 아쉬움이 남습니다.

Ensemble은 soft voting도 해보았고 hard voting도 해보았는데 의외로 hard voting의 성능이 더 좋았습니다. 비록 model parameter 저장도 뒤늦게 시작하긴 했지만, 아무튼 지금까지 만들어놓은 각종 모델들로 soft voting, hard voting을 하면서 성능이 꽤 많이 올랐고 더 이상 오르지

않는 성능에 쌓여가던 지루함이 여기서 많이 풀렸던 것 같습니다. hard voting도 여러 방법이 있다는 것을 여기서 알 수 있었습니다. 성능이 좋은 모델에 가중치를 더 높게 부여하여 hard voting을 하는 방식, ensemble로 만들어진 결과를 다시 다른 모델과의 ensemble에서의 vote 요소로 활용하는 방식 등 여러 방법들을 활용해보았습니다. 쌓아놓은 모델이 많다보니 ensemble을 적용할 당시에는 제출기회가 부족해질 지경이었던 기억도 납니다. ㅋㅋㅋ

ver. 5	**Model** Backbone EfficientNet B0 / ResNeXt 50 Classifier FC layer X3 (3 class/2 class/3 class) return x, y, z **Criterion – CrossEntropyLoss** **Optimizer – Adam** **Scheduler – ReduceLROnPlateau** **Augmentation –** **CenterCrop, RandomBrightnessContrast, HorizontalFlip, CLAHE, ShiftScaleRotate**

역시 multi-head 모델이지만 앞선 v3과의 차이점은 각 category별로 따로 학습을 시키지 않았다는 점입니다. 이렇게 하면 역전파가 동시에 전달되므로 어느정도 검증된 parameter를 얻을 수 있다는 장점이 있습니다.

계속해서 v4를 어떻게 개선할지에 대해 고민하던 찰나, 토론 게시판에 올라온 글을 보고 뒤늦게 노선을 변경하였고 꽤 놀라운 성능을 보여주어 바로 이 모델을 주 모델로 채택하였습니다.

사실 앞서 언급했듯이 v4에서 대부분의 시간을 사용했습니다. 그런데 거의 시간을 쓰지 않은 v5에서 오히려 더 좋은 성능을 얻을 수 있었습니다. 하지만 v4에서 여러 방법들을 실험보았기 때문에 여기서는 loss function, optimizer, scheduler, augmentation 등을 더 이상 변화시키지 않고 learning rate 및 backbone 모델만 조정하면서 편하게 여러 실험들을 해볼 수 있었습니다. 특히 minor data만 augmentation시키는 것이 v5에서 꽤 큰 성능 향상을 보여주었습니다.

싱글 모델의 성능이 잘나오다보니 앙상블 성능도 역시 잘나왔습니다. 다만 여기서는 hard voting보다 soft voting이 더 성능이 좋아 soft voting을 활용하였습니다. v4에서는 왜 hard voting이 더 좋았는지는 아직까지 좀 의문입니다. age 모델을 분리하면서 벌어진 현상 같은데 age만 예측하는 것이 뭔가 예상치 못한 결과를 불러왔을까요..? 그래도 soft voting이 무조건 좋다라는 제 고정관념을 깰 수 있었던 것 같아 이 역시도 꽤 의미있는 결과였던 것 같습니다.

앞서 언급했듯이 KFold를 적용하지 못했던 점이 아직까지 아쉬움이 남습니다. 마지막 제출에서는 valid data ratio를 0.002로 주고(사실상 없게 하기 위해) 학습을 돌린 후 제출하였는데 KFold로 만든 모델을 ensemble했다면 이보다 더 좋은 성능이 나오지 않았을까하는 생각이 듭니다. 물론 이번 competition을 진행하면서 '무조건은 없다'라는 것을 깨달았기 때문에 이것은 어디까지나 작은 아쉬움일 뿐입니다. ㅋㅋㅋㅋ

최종 submission에 활용한 모델 및 전략

Model(ver. 5)
Backbone
EfficientNet B0 / ResNeXt 50
Classifier
FC layer X3 (3 class/2 class/3 class)
return x, y, z

Criterion – CrossEntropyLoss
Optimizer – Adam
Scheduler – ReduceLROnPlateau
Augmentation –
CenterCrop, RandomBrightnessContrast, HorizontalFlip, CLAHE, ShiftScaleRotate

Epochs: 12 / **Batch_size**: 16 / **lr**: 3e-05 / **val_ratio**: 0.002 / **lr_decay_step**: 5
Ensemble(Soft voting) EfficientNet B0 + ResNeXt50
기타 전략: 노년층 데이터만 추가적으로 리샘플링 및 증강

마주한 한계와 도전 숙제

- 텐서보드나 wandb 등의 시각화 툴을 활용하지 못했다는 점이 아쉽습니다. 이를 적용하기 위해서는 모든 코드를 뜯어 고쳐야하는 것이 아닌가하는 두려움과 귀찮음에 시도조차 해보지 않았는데 주말에 시도해보아야 할 것 같습니다.
- iPython 파일에서 IDLE 기반 프로젝트로 환경을 바꾸면서 KFold 검증 전략을 사용하지 않게 되었습니다. 다만 앞서 언급했듯이 valid set을 0.0으로 아예 주지 않는 것보다 KFold 5개로 ensemble을 하는 것이 더 성능이 좋았을 것 같습니다.
- 이번 competition을 진행하면서 data augmentation 방법, hyperparameter, criterion, optimizer, scheduler 등 많은 것을 바꿔가면서 모델을 학습시켰는데 정작 모델 구조를 어떻게 짤지에 대한 생각을 깊게 하지 못했던 것 같습니다. 여기엔 v1부터 v5까지 길게 적어놨지만 대부분의 시간을 v4에 쏟았으니까요. 다음 competition부터는 모델 구조를 어떻게 잡을지에 대한 깊이 있는 생각도 해보아야 할 것 같습니다.