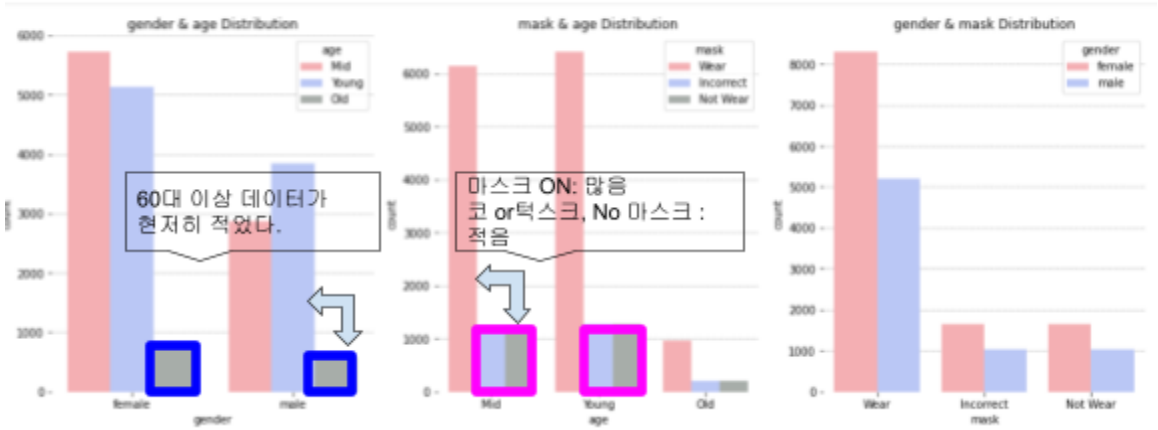
신문종

public : 85위 acc: 80.2222% F1 score: 0.7420

private: 75위 acc: 79.6571 F1 score: 0.7377

문제점

1. 굉장히 심한 데이터 불균형



출처: [EDA를 통해서 얻은 인사이트 공유해보기 by 오혜린님]

2. 18개의 비슷한 다중 클래스 분류

ImageNet대회의 경우 비행기, 자동차, 노트북 등 특징이 뚜렷한 클래스를 분류하는 반면 사람 얼굴, 마스크 여부, 나이는 그 차이가 뚜렷하지 않다.

따라서 18개의 클래스를 한번에 예측하는것보다 나이, 성별, 마스크 착용 여부를 각각 판단하여 예측하는것이 더 나은 성능을 가져올것이라 예측했다.

모델 학습

첫번째 시도:

학습하는 연습을 하기위해 아무런 파라미터 서칭 없이 무작위로 랜덤하게 집어넣었다.

Dataset을 torchvision의 ImageFolder로 전부 디렉토리에 따로따로 넣어준 뒤 예측을 수행하였다. 이때 클래스와 인덱스가 일치하지 않아 예측에서 값이 엇나가는 문제가 있었지만 custom ImageFolder dataset을 구현하여 해결하였다.

(http://boostcamp.stages.ai/competitions/1/discussion/post/18 by 고지형님)

LB점수: acc : 72.7600% f1 : 0.62

Backbone: Resnet101

optimizer: Adam

Learning rate : 0.001

img_size = 224 X 224

Data augmentation : 적용하지않음

18개 클래스 예측

두번째 시도:

마스크를 쓰고있지 않은 데이터가 현저히 적다고 느껴 외부 데이터셋을 가져와 학습을 진행

평가지표가 F1 score로 바뀌어 F1 loss와 데이터 분포를 loss로 학습하는 Crossentropy를 사용

(마스크를 쓰고있지 않은 아시아인 potrait 추가)

LB점수: acc : 77.25% f1 : 0.7274

Backbone: Resnet101

optimizer: Adam

Learning rate : 0.0001

img_size = 224 X 224

Data augmentation : 적용하지않음

18개 클래스 예측

extra dataset 추가

세번째 시도:

한번에 18개 클래스를 구분하는 task는 feature별로 예측하는것보다 비효율적일것이라 생각하여 3-hot encoding으로 클래스를 구분

(피어세션 T1200조원 님의 말씀대로 구현해보았다)

해당 인덱스로 라벨링을 구현

[남자 여자 <30 >=30 & <60 <60 correct incorrect not_wear]

[성별 성별 나이 나이 나이 마스크 마스크 마스크]

ex) 남자 30세미만 마스크를 쓴 사람이라면

[1 0 1 0 0 1 0 0]

LB점수: acc : 79.3175% f1 : 0.7327

Backbone: Resnet101

optimizer: Adam

Learning rate : 0.0001

img_size = 224 X 224

Data augmentation : 적용하지않음

마지막 Linear layer : 18 -> 8

3 hot encoding으로 각 feature 예측

extra dataset 추가

네번째 시도:

더 학습을 하기엔 시간이 부족하여 지금껏 학습한 모델들의 csv파일을 읽어 voting

LB점수: acc : 80.2222% f1 : 0.7420

csv 파일 불러들여 row별 최빈값을 이용

어려웠던점

torchvision의 ImageFolder

첫 4일은 디렉토리별로 이미지를 넣어놓으면 클래스별로 데이터셋을 만들어주는 ImageFolder를 이용하여 학습을 진행하였다.

학습을 진행하고 제출내역을 확인하였을 때 1-20%의 정확도 점수가 나와서 문제점을 찾고있었다.

분류한 디렉토리를 전부 삭제하고 처음부터 다시 나눴지만 유의미한 결과는 없었다.

class_to_idx method에 있는 딕셔너리만 바꿔주면 될 줄 알았지만 해결되지 않았다.

3월 31일 토론게시판 고지형님이 올려주신글을 보고 Dataset구축, 72% acc 달성

Data augmentation

Centercrop, Rotation을 적용하고 학습을 진행했을때 역시 50%대의 정확도가 나왔다.

아무것도 추가하지 않은 베이스라인이 70%대의 정확도를 보여준 반면 데이터를 증강 시켰을 때 오히려 정확도가 떨어졌다.

피어세션에서도 같은 이야기가 자주 등장했다.

문제는 Centercrop이었던것 같다.

데이터셋에 얼굴이 꽉 들어찬 사람들의 사진이 있었으며 이사람들의 사진을 crop하면 눈, 코 등이 사라지는 현상이 발생한다.

몇몇 캠퍼들은 사진에 패딩을 적용 후 crop을 적용하여 문제를 해결한 사례도 보았다.

Tensor board와 **nni**

이번 대회는 데이터셋으로 인한 삽질(?)이 절반 이상을 차지했다.

많은 실험을 진행하지 못했고 하이퍼 파라미터 역시 감으로 설정하였다.

특히 아쉬운점은 Tensor board와 nni를 사용하지 못했다는점이 정말 아쉽다.

모델이 어떻게 학습하고있고, 정확도가 어떤 추세로 좋아지고있는지 체크하지 못했다.

앙상블

단순히 내가 제출한 파일중 정확도 70% 이상 F1 0.7이상의 파일을 받아 하드보팅하는 형식으로 적용했다.

이 방식의 앙상블은 반쪽짜리라 생각한다.

각각의 모델에서 나온 inference확률을 기반으로 soft voting을 적용하지 못한것이 아쉽다.

또한 단순 하드보팅으로도 어느정도 성적을 끌어올릴 수 있다는 사실에 앙상블의 힘을 느낄 수 있는 계기이기도 했다.

느낀 점

베이스라인 제작

고생해주신 멘토분들께 조금은 죄송한 이야기지만 대회 기간에 .py로 제작된 베이스라인을 보지 않았다.

스스로 먼저 만들어 본 뒤 대회가 끝나고 비교하며 이런점이 아쉬웟고 저런점은 잘한것같다 라는 것을 느끼려는 의도였다.

베이스라인을 보고 코드를 더 정교하고 재생산성이 높게 구현했다면 성적이 조금 더 높게 나왔을 수 있겠지만 모델구성, 학습, 데이터셋을 스스로 원하는 방식으로 구현해보았다는것에 스스로 플러스 점수를 주고싶다.

피어세션과 지식 공유의 중요성

모델 성능을 향상시키는 아이디어들은 대부분 피어세션에서 얻었다.

운이좋게(?) 매번 상위권에 랭크된 캠퍼분들을 많이 만날 수 있어서 그분들에게 여러가지 방법론과 적용 아이디어 등을 물어보았고 순위를 올릴 수 있었다.

다음 대회부터는 토론게시판에 글도 올리고 받은만큼 베풀어야겠다는 생각이 들었다.

IDE와 **CLI**를 다루는 능력의 향상

pycharm IDE에 나만의 Live template을 만들어 코드를 빠르게 만들 수 있었고 CLI에서 tmux를 사용하여 병렬적으로 학습, 관리, 실험 등을 할 수 있었다.

주피터 노트북으로 작성을 하게 된다면 추후 코드 관리도 불편할 뿐더러 여러가지 실험을 한 흔적이 하나의 노트북에 몰려있어 중구난방식으로 코드를 짜게되어 어떤 노트북에 어떤 코드가 들어있는지 파악하기 힘들어진다.

하지만 .py형태로 작성을 하려면 보다 모듈화된 방식으로 코드를 작성하기 때문에(거의 반 강제적) 재사용성이 굉장히 늘어났다.