Project1 Warp up Report

Title	Type	Date	Name
마스크 착용 이미지 분류	개인 프로젝트	2021.03.29 ~ 2021.04.08	T1138 윤준호

Result

- Leaderboard
  - F1 score: **0.7235**
  - Rank: **109**/224

Task

- 4500명 x 7장 image data
- 18 classes multi-label classification
  - 마스크 착용 여부(3), 성별(2), 나이(3) 기준

- **Main Issues**
  - Data imbalance
    - 클래스 조건별 imbalance 존재, 특히 '나이'에서 '60세 이상' class에 해당하는 데이터가 7% 정도로 매우 적음
  - Noisy label
    - 잘못 labelling된 데이터 존재 (최소 10장 이상)
  - Dependent classes
    - 독립된 18개의 class가 아니라, 세 가지 기준에 의해 경우의 수로 나뉜 class들이기 때문에 독립분포 가정 위배

Approach

Dealing with Data Imbalance

1. Age class의 imbalance를 완화하기 위해 Age class 기준점을 변경
   - 변경 전: 30 미만 ,30-59, 60 이상
   - 변경 후: 30 미만, 30-57, 58 이상
2. 예측 확률이 낮은 class에 대해 더 높은 loss를 부여하는 Focal loss 사용
   - Default 값이 2인 gamma를 0.8 까지 낮춰, loss 가중치를 너무 과하지 않게 부여함 (gamma=0일 때 CrossEntropy와 동일함)

Generalization

1. Train-validation set을 사람을 단위로 split
   - 같은 사람의 데이터가 train, validation set에 각각 들어가 validation accuracy가 과대평가되는 것을 방지
2. Light한 모델 사용
   - Overfitting 되는 것을 방지하기 위해, 매우 가벼운 EfficientNet b0로 시작해 한 단계씩 높여가며 실험 진행 (최종적으로는 가볍다고 하기 애매한 b4까지 가긴 함)
3. Early stopping
   - Train loss와 validation loss를 텐서보드를 이용해 그래프로 비교해가며, train loss는 계속해서 감소하는데 validation loss가 다시 증가하는 구간에서 학습 중단
4. Data augmentation
   - 여러가지 transformation을 활용해 데이터의 state, case를 다양화
   - 여러 실험을 거쳐서 성능 향상에 도움이 되는 transformation만 남김 (centercrop, resize, normalize)
5. Batchsize 최대화
   - Batch normalization 단계에서 최대한 일반화된 분포를 기준으로 normalization할 수 있도록, GPU 용량과 성능이 감당하는 한에서 최대화 (batchsize=128)
   - Resize를 많이 주어 batchsize를 더 올릴 수 있게 함
6. AdamW optimizer 사용
   - 수렴이 빠르고 대체적으로 다양한 task에서 높은 성능을 보이는 Adam의 단점으로, SGD에 비해 낮은 일반화 성능이 언급됨 (L2 regularization이 Adaptive optimizer에서 일반화 효과가 적어짐)
   - Adam의 일반화 성능을 높이기 위한 방법으로, weight update 시 weight decay를 적용하는 AdamW를 사용함
7. Pseudo Cross-validation
   - 마지막 제출 시 validation set 없이 train set 100%로 학습 진행
   - 최적의 epoch를 max epoch로 지정 후 최종 모델을 활용해 제출

Model & Hyperparameters

- Model
  - EfficientNet b4
- Hyperparameters
  - Optimizer: AdamW
  - Loss: Focal loss
    - gamma: 0.8
  - Learning rate: 3e-4
  - LR scheduler: StepLR
    - stepsize=2
    - gamma=0.2
  - Epoch: 4
  - Batchsize: 128
  - Transform
    - RandomRotation(10) (train only)
    - CenterCrop((450, 360)) (train only)
    - Resize(200, 200)
    - Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])

Lessons

Failures

- Oversampling
  - weighted ramdom sampler를 사용해 시도했지만, 오히려 f1 score가 떨어짐
  - class 별로 적절하게 중복 sampling할 비율을 지정하는 방식이 아니라, class 별 비중을 계산한 뒤 모든 데이터에 대해서 weight를 계산한 후 거기 맞춰서 random sampling하는 방식
  - 이번 task에서는 data가 지나치게 imbalance 되어 있었기 때문에, 강제로 1 : 1 : 1 : … 비율로 맞추면 오히려 역효과가 나는 것 같음
  - 실패 후에 비슷한 목적으로 이용 가능한 focal loss를 시도해 성능 향상을 보았음
  - class 별로 지정한 비율로 sampling할 수 있도록 구현한다면 성능 향상에 도움이 될 수도 있을 것 같음
- Grayscale, horizontal flip
  - 너무 단순하게도 "다양한 색의 마스크를 헷갈리지 않게 잘 분류하도록" grayscale을 전부 적용했지만, 오히려 transforms에서 제외했을 때 성능이 더 좋았음
  - validation, test set의 transfroms에 포함할 때와 하지 않을 때 모두 마찬가지였음
  - Hyperparameters를 공개한 규진님의 토론 게시물을 봤는데, transform을 전혀 사용하지 않은 것을 보고 충격 받고 insight를 얻어서 실험 후 발견함

Untried Ideas

- 1 backbone + 3 branches model
- 외부 데이터셋 활용 or 새로운 데이터셋 생성
- Soft-f1 loss

Thoughts

> 무엇보다도 Dataset class와 Data loader를 온전히 내 힘으로 task에 알맞게 구현해내고, 계획한 아이디어에 맞게 아키텍처를 구성해 end-to-end 모델을 완성했다는 것이 정말 뿌듯하다.
>
> 직접 밑바닥부터 코딩을 해보면서 왜 딥러닝 framework들이 그렇게 OOP를 중요하게 생각하는지 몸소 체험할 수 있었고, 새로운 아이디어를 적용할 때 내 custom class들에 맞게끔 내부 로직을 수정하는 요령도 생겼다.
>
> 다양한 아이디어를 적용하고 실험해보면서, "대개 이런 경우에 이런 방법이 있다"고 하는 ML/DL 지식이 생각보다 그대로 적용되는 경우가 잘 없다는 것을 느꼈다. 앞으로는 실험을 할 때 조금 더 체계적인 tool과 logging을 사용하면서, 정확히 하나의 변수만 변화시켜가면서 진행해야겠다. 또한, 크게 성능 향상에 기여할 것 같지 않다고 생각되는 아이디어들도 그냥 지나쳐버리지 않고, 작게나마 실험을 구성해 직접 검증한 뒤에 판단해야겠다.