P-Stage 1 Image Classification 대회 개인 회고

T2234 최현진

1. 프로젝트 목표

이번 프로젝트 가장 큰 목표는 실제 주어진 Task를 해결하기 위해 그동안 배웠던 다양한 기법을 최대한 많이 적용하고 실험하여 경험을 쌓는 것과 전체 학습 및 추론 과정을 Jupyter notebook 환경이 아닌 소스코드 구현을 통해 ML Engineer의 역할 중 일부를 경험하는 것입니다.

2. 목표 달성 계획

- 개인 학습

  데이터 전처리, Transfer learning, Data Augmentation, Hyperparameter tuning 등 P-Stage 이전에 학습했던 내용 복습 및 적용

  MLOps Pipe line에 대한 전반적인 지식을 학습하여 실무에서 Engineer 업무 파악

  학습 및 추론 단계를 자동화할 수 있는 프로그램을 개발하기 위해 Pytorch-Template을 참고하면서 clone coding, 프로그램의 구조, 동작 흐름을 학습

- 공동 학습

  문제 해결을 위한 다양한 접근 방법 공유

  이미지 분류에 대한 여러 기법 Keyword 공유 및 학습

  협업을 위한 git 사용방법 숙지

3. 모델 개선 방안

- Model Search

가장 성능이 좋은 모델을 찾는 것이 우선이라 생각하여 단순히 Pre-trained 모델 중 제출했을 때 점수가 가장 좋은 모델을 탐색한 뒤 Hyperparameter tuning 적용하였습니다.

- Class Imbalance

데이터셋의 각 클래스별 분포를 확인하여 상대적으로 부족한 클래스의 양을 증가시키는 Oversampling 기법을 생각해보았습니다. 특히 60대 이상의 데이터가 부족하였는데 원본 이외 4~5장을 복제하여 transform을 random하게 적용해 데이터셋을 증가시키는 방법으로 적용했습니다. Overfitting을 방지하여 일반화된 모델을 만들 수 있는 기대하였습니다.

- Bias and cheating

일반화가 잘된 견고한 모델을 만들기 위해 bias를 제거하고 cheating 현상을 방지해 최종 점수를 높일 수 있는 방법을 생각해 보았습니다. torch가 지원하는 다양한 transform 함수를 사용해 augmentation을 적용하였습니다. 추가적으로 MTCNN 라이브러리를 사용해 얼굴을 detection 하고 crop한 데이터셋을 따로 만들어 bias를 제거하려고 시도하였습니다.

- Hyperparameter Tuning

모델, 전처리, Seed를 고정하고 learning rate, optimizer, batch size 등 Hyperparameter를 변경하면서 모델 성능을 끌어올리기 위한 방법을 시도하였습니다.

4. 행동 결과 및 깨달음

- Model

여러가지 pre-trained 모델을 사용해본 결과 모델의 깊이와 parameter 크기는 학습 결과에 큰 영향을 주지는 않았습니다. 동일한 조건에서 가장 성능이 좋았던 EfficientNet-b4을 사용하였고 성능을 올리기 위해서는 데이터 분석과 전처리 과정이 중요하다는 것을 깨달았습니다. 또한 transfer-learning을 할 때 fine tuning이 성능이 좋았습니다.

- Class Imbalance

불균형 문제를 해결하기 위해 Loss 함수에 weight 벡터를 적용하거나 부족한 클래스의 데이터셋 Oversampling을 적용하였지만 효과가 없었습니다. 프로젝트가 끝나고 다른 팀원분들과 이야기를 해보니 잘못 라벨링된 데이터가 있었습니다. 문제가 있는 클래스를 직접 관찰하는 것도 방법이 될 수 있다는 것을 깨달았습니다.

외부 데이터를 사용해서 부족한 클래스를 보충하는 방법도 효과가 있다는 이야기를 듣고 다음에 이런 상황에서 적용해 볼 수 있는 방법이라는 것을 배웠습니다.

다양한 loss function (Label Smoothing loss, Focal loss, F1 loss)을 적용해 보았습니다. 특히 Focal loss를 적용 했을 때 점수가 상승하였는데, 불균형 데이터셋에 적용할 때 효과적인 함수인 것 같습니다.

- Bias and cheating

모델의 cheating 현상을 방지하여 견고한 모델을 만들고 점수를 높이는 의도로 시도했던 방법이었으나 오히려 점수가 하락했습니다. 프로젝트가 끝나고 생각을 해보았는데 데이터의 특징을 완전히 지우는 요인이 되었다고 생각하였습니다.

- Hyperparameter tuning

Hyperparameter tuning은 시간도 오래 걸리고 성능 향상도 크지 않았습니다. 목표한 성능

에 어느정도 도달하면 그 때 시도해볼 방법인 것 같습니다.

5. 새롭게 시도한 변화 및 효과

- Model 분석

  성능이 좋지 않은 모델이 현재 어떤 문제가 있는지 정의하기 쉽지 않았습니다. 멘토링을 통해 모델의 문제점을 분석하는 방법 중 하나인 Confusion matrix에 대해 알게 되었고 적용해보니 현재 model의 문제를 한눈에 파악할 수 있었습니다. 그 후로는 그 문제를 해결할 수 있도록 집중할 수 있었습니다.

- Multi Model

  마스크 착용상태, 성별, 나이 (3 * 2* 3)를 한 번에 18개 클래스로 예측하는 모델로 생성됩니다. 각 task에 대해 예측하는 모델 3개(mask, gender, age)를 만들었을 때 결과가 더 좋았습니다.

- Regression

  나이를 예측하는 모델을 regression 문제로 접근하였습니다. 모델이 나이를 추측할 수 있도록 output을 1로 설정하여 모델을 설계하였고 추측 결과를 후처리 하는 과정에서 confusion matrix가 가장 이상적인 모양이 되도록 값을 기준으로 다시 3개의 범위로 나누어 결과를 도출하였습니다. 그 결과 검증 데이터에 대한 f1 score는 상승하였으나 평가 데이터에 대한 제출 결과는 좋지 않았습니다. 후처리 하는 과정에서 경계를 분할하는 값이 평가데이터에는 이상적이지 않았다고 생각합니다.

- K-Fold Out-Of-Fold

  Fold 단위로 나누어진 데이터셋을 각각의 모델들이 학습하고 평가하는 방식을 사용하였을 때 가장 점수가 좋았습니다. 무엇보다 결국 전체 데이터셋을 학습하는 것이 최종 점수 상승으로 이어졌다고 생각합니다. 상대적으로 데이터셋이 부족한 상황에서 적용해 볼 수 있을 것 같습니다.

- Early Stopping

  Overfitting이 발생하기 전에 학습을 종료하여 검증 데이터에 대한 점수가 상승하였고 제출 점수도 상승하였습니다.

  검증 데이터에 대한 정확도가 더 이상 오르지 않으면 학습을 조기 종료할 수 있어 시간을 단축 할 수 있었습니다.

6. 한계 및 아쉬웠던 점

- Trial and error 방식으로 접근

초기 모델 선정 단계에서 체계적인 실험을 하지 못한 부분이 가장 아쉬웠습니다. 데이터셋에 대한 충분한 분석을 하지 않고 여러가지 Pre-trained 모델을 단순히 사용해 보면서 가장 성능이 좋은 모델을 선정하는 일이 제대로 이루어 지지 않았습니다.

- seed 고정

  초기에 seed를 고정하지 않고 실험을 했던 시간이 꽤 많았습니다. 그로 인해 체계적이고 명확한 실험을 진행하지 못하였습니다. 실험 단계에서 재현을 위해 정말 중요한 작업이라는 것을 많이 배웠습니다.

- Data Augmentation

  Cutmix, face crop 등 다양한 Augmentation을 적용해보지 못한 부분이 너무 아쉬웠습니다. 시간이 부족했던 물리적인 한계가 있었지만 이번 프로젝트를 계기로 효율적인 방법을 배워 다음 프로젝트에서는 다양하게 적용해 보고 싶습니다.

- Ensemble

  성능이 좋았던 모델들의 추론 결과를 평균을 내어 최종 결과로 사용하는 방식을 사용하였습니다. 제출 결과 성능 상승으로 이어지진 않았지만 앙상블이 효과가 없는 것이 아닌 사용된 모델들의 문제인 것 같아 아쉬웠습니다.

7. 다음 P-Stage에서 시도해볼 것

- 데이터 분석 단계에서 다양한 분석 기법을 적용해 보는 것
- 효율적으로 실험을 할 수 있도록 자동화 소스코드를 좀 더 일찍 작성하는 것
- 모듈화가 잘 이루어진 소스코드를 API로 활용해보기