2023 데이터 크리에이터 캠프

DATA CREATOR CAMP

starter

고등부

1. MISSION1

1-1

- training, test 데이터 이미지 수 : 129600, 16200

os.walk를 사용하여 현재 디렉토리 경로, 모든 디렉토리 이름 담은 리스트, 모든 파일 이름 담은 리스트 반환하고 디렉토리 내의 모든 파일과 하위 디렉토리를 모두 순환하여 len()으로 모든 파일 수를 세고 sum()으로 합한다

1-2

- Training, Test 01, 02, 03 이미지 개수 : 43200, 43200, 43200 / 5400, 5400, 5400

01, 02, 03에 대한 리스트를 만들고 os.walk를 사용하여 모든 파일이름 담은 리스트를 반환, '_' 기준으로 나누고 마지막이 01, 02, 03일 때 append를 사용하여 각각의 리스트에 저장한다

2. MISSION2

2-1

이미지축소크기:64*64

- 이미지 크기를 줄이면 픽셀수가줄어들어 학습에 필요한 계산량이 줄어 학습 속도를 빠르게 할수 있다.
- 메모리에 로드되는 데이터의 크기가줄어들어 메모리 사용량을 줄일 수 있다.
- 과적합을 감소시킬 수 있다.

2-2

(2)이미지를 하나의 압축파일로 만들어 colab으로 전송

- 파일 크기가줄어들어 데이터 전송 시간과 저장공간이 줄어든다.
- 파일이 손상되거나 유실되는 것을 방지할수 있다.
- 파일의 이동,복제,삭제 등의 관리가편리하다

3. MISSION3

3-1

마스크 착용여부 이진분류 실행

?< 이진분류 수행 순서
>

라벨링 -> 파일분류

CNN 모델 구성

손실함수 / 최적화 알고리즘 설정

이미지 데이터 제네레이터 생성

콜백 설정

모델 학습

< 분석 내용 >

- Image Data Generator 클래스 사용해 학습 / 검증을 위한 두 개의 이미지 데이터 제네레이터 생성
- Sequential 클래스 사용해 CNN 모델 생성
- model.compile ( ) 메서드사용해 ‘adam‘ 최적화기, ‘binary_crossentropy‘ 손실함수 선택 / ‘accuracy‘, ‘f1_score‘ 를 모델성능 평가하는 지표로 사용하도록 모델 컴파일
- 학습 데이터 / 검증 데이터로 앞에 생성한 이미지 데이터 제네레이터를 사용하여 model.fit ( ) 메서드를 이용해 모델 학습

3. MISSION3

- 3-1
- Test 데이터의 f1 score 값 : 0.6643682428974251

-model.predict(val_gen) 이용해 검증 데이터셋에 대한 예측값 생성

-이진분류변환-검증 데이터셋의 실제 라벨 값을 정수형으로 변환

-TensorFlow의 Precision, Recall 클래스사용하여 예측값, 실제 라벨 비교해 정밀도/재현율 계산, updata_state() 메서드 이용해 각 메트릭 객체의 상태 업데이트

-계산된 정밀도와 재현율 이용해 F1 점수 계산

3. MISSION3

3-2

클래스 불균형 문제 확인 & 문제 해결

<클래스 불균형 문제 확인>

학습 데이터 - 미착용 : 43200 개

학습 데이터 - 착용 : 86400 개

검증 데이터 - 미착용 : 5400 개

검증 데이터 - 착용 : 10800 개

<클래스 불균형 문제 해결 : 언더샘플링>

언더샘플링으로 다수 클래스의 샘플을 제거함으로써 데이터의 크기를 감소 할 수 있다.

클래스 샘플이 많아 학습에 방해 될 수 있음으로 다수 클래스의 샘플을 제거해 과적합을 방지 할 수 있다.

- os.makedirs() 함수 사용해 언더샘플링 데이터 저장할 디렉토리 생성, exist_ok=True 설정해 디렉토리가 이미 있어도 오류가 발생하지 않게 함
- 각 클래스 이미지 파일 목록 가져와서 가장 적은 이미지 파일을 가진 클래스 데이터 수 구해서 이 기준으로 언더샘플링된 학습 데이터 디렉토리에 복사
- 동일하게 검증 데이터 언더샘플링

3. MISSION3 추가 개선 사항

3-2

클래스 불균형 문제 확인 & 문제 해결

<클래스 불균형 문제 확인>

학습 데이터 - 미착용 : 43200 개

학습 데이터 - 착용 : 86400 개

검증 데이터 - 미착용 : 5400 개

검증 데이터 - 착용 : 10800 개

<클래스 불균형 문제 해결 : 오버샘플링>

오버샘플링으로 언더샘플링보다

- os.makedirs() 함수사용해 언더샘플링 데이터 저장할 디렉토리 생성, exist_ok=True 설정해 디렉토리가 이미 있어도 오류가 발생하지 않게 함
- 각 클래스 이미지 파일 목록 가져와서 가장 적은 이미지 파일을 가진 클래스 데이터 수 구해서 이 기준으로 언더샘플링된 학습 데이터 디렉토리에 복사
- 동일하게 검증 데이터 언더샘플링

3. MISSION3

3-2

Accuracy, Pricision, Recall, f1 score 계산

Accuracy : 0.5047222375869751

Precision : 0.5047004818916321

Recall : 0.5070370435714722

F1 Score : 0.5058660409304978

- Accuracy, Precision, Recall 클래스의 인스턴스 생성
- 데이터 제너레이터의 상태 초기화
- 검증 데이터에 대한 모델의 예측 값 구함
- 예측값을 0.5 기준으로 0또는1로 변환
- 실제 라벨값을 문자열에서 정수로 변환
- update_state 메소드 호출해 실제 라벨 값과 예측 값을 받아 메트릭의 계산을 업데이트
- result, numpy 메소드 호출해 메트릭 결과 numpy 배열로 변환
- 정밀도, 재현율 결과 이용해 F1score 계산

(3-1에서 F1Score 에 비해 3-1에서 F1Score이 줄어들었다)

3. MISSION3

- 3-3
- 성능 향상 시도 과정 / 정확도 변화 - 8번 시
  도

1 - 정확도를 비롯한 점수가 불균형 문제 해결 전보다 줄어듬

2 - 드롭다운과 Batch Normalization를 같이 진행
-> 정확도 / 정밀도 / 재현율 / f1 점수가 증가

3 - 두번째 시도에서 드롭다운의 비율을 0.3으로 줄임
-> 정확도 / 정밀도는 그대로, 재현율 / f1 점수는 줄어듬

4 - 드롭다운의 비율을 0.7로 올림
-> 정확도 / 정밀도는 향상하기 전과 같고, 재현율 / f1 점수는 증가 (성능 조금 향상)

3. MISSION3

- 3-3
- 성능 향상 시도 과정 / 정확도 변화 - 8번 시
  도

5 - 성능개선이 없을 때 학습이 멈추는 patience 의 수치를 3에서 5로 증가시킴
-> 모든 점수가 대체적으로 줄어듬

6 - patience 수치를 다시 3으로 바꾸고 배치 크기 (한번에 학습하는 수) 를 64로 증가시킴
-> 재현율을 제외한 모든 점수가 조금씩 증가

7 - 배치크기를 다시 32로 하고 손실함수 변경
-> 정확도 / 정밀도 떨어짐, 재현률 / f1 점수 증가

8 - 손실함수를 focall loss 로 바꿈
-> 정확도 / 정밀도는 성능향상 전과 동일, 재현율 / f1 점수는 감소

3. MISSION3

3-3

결과

6번째 수행 과정에서 재현율을 제외한 모든 점수가 증가해 성능이 가장 좋았다

이런 결과가 생긴 이유는 배치크기를 늘렸을 때 더 안정적이고 최적화 과정에서 노이즈를 줄여주었고 GPU의 병렬 처리 능력을 더욱 활용할 수 있게 해주어 훈련 속도를 높이고, 결과적으로 성능 향상에 도움을 주었기 때문이다.

3-4

왜 오류가 났을까?

오류가 난 사진들을 보았을 때 대부분이 마스크 미착용 클래스였다.

마스크 미착용 클래스에서만 오류가 난 이유로
먼저 학습률이 마스크 미착용 클래스의 패턴과 맞지 않다는 것과, 학습속도를 올리기 위해 데이터의 품질을 낮춘 것 때문이라고 생각한다.

감사합니다.

고등부 **starter**

팀장 오윤찬 박성현 최성욱 김시연

2023 데이터 크리에이터 캠프

DATA CREATOR CAMP