**2024년 1학기 인공지능기초 기말고사**

학과: 소프트웨어학과 학번: 202020827 이름: 김경민

 보고서:

1. **개발 환경 설정 (사용한 라이브러리 버전 명시)**
2. 필요한 라이브러리설치
3. !git clone https://github.com/ultralytics/yolov5.git
4. %cd YOLOv5
5. %pip install -r requirements.txt
6. %pip install opencv-python-headless

**사용한 라이브러리 버전**

Python version: 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]

PyTorch version: 2.3.1+cu118

NumPy version: 1.26.4

Pandas version: 2.2.2

OpenCV version: 4.8.1

Matplotlib version: 3.9.0

Scikit-learn version: 1.5.0

YAML version: 6.0.1

1. **모델 훈련 과정 설명 및 결과 분석 (손실 함수 변화 그래프, 성능 지표 포함)**

#훈련 명령어

!python train.py --img 640 --batch 16 --epochs 5 --data dataset.yaml --weights yolov5s.pt

모델훈련과정

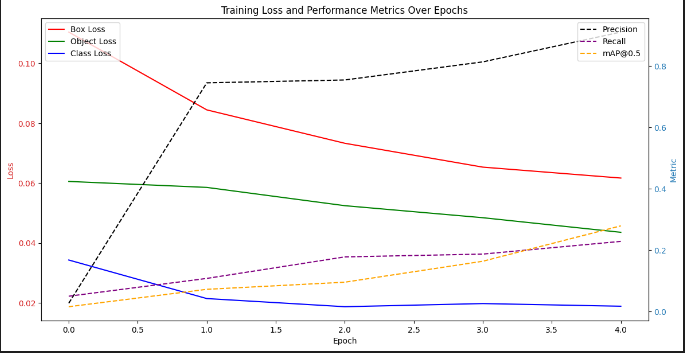
--img 640: 입력 이미지의 크기를 640x640 픽셀로 설정한다.

--batch 16: 배치 크기를 16으로 설정하여, 각 반복에서 16개의 이미지를 동시에 처리한다.

--epochs 5: 총 5회의 에포크(epoch) 동안 모델을 훈련한다. 에포크는 전체 데이터 세트를 한 번 통과하는 것을 의미합니다.

--data dataset.yaml: 데이터셋 구성을 정의한 YAML 파일을 사용합니다.

--weights yolov5s.pt: 사전 훈련된 가중치를 초기 가중치(0,01)로 사용하여 훈련을 시작한다.



**그래프 분석**

**손실 함수 변화**

1. **Box Loss** (빨간색): 초기에 가장 높은 손실 값을 보이며 0.1 부근에서 시작하여 급격히 감소, 이후에는 완만한 감소세를 보입니다. 이는 모델이 객체의 바운딩 박스를 정확하게 예측하는 능력이 점진적으로 향상되고 있음을 나타낸다.
2. **Object Loss** (녹색): 상대적으로 안정적인 감소 추세를 보이며, 손실이 지속적으로 줄어들고 있다. 이는 모델이 객체의 존재 여부를 판별하는 데 점점 더 능숙해지고 있음을 의미합니다.
3. **Class Loss** (파란색): 비교적 낮은 시작점에서 점진적으로 감소하며, 이는 모델이 객체의 클래스를 정확하게 분류하는 능력이 개선되고 있음을 보여준다.

**성능 지표**

1. **Precision** (**검은색 점선**): 에포크가 진행됨에 따라 점진적으로 상승하는 경향을 보인다. 이는 모델이 실제로 정확한 예측을 늘려가고 있음을 의미하며, False Positive의 수가 감소하고 있음을 나타낸다.
2. **Recall** (보라색 점선): 성능이 일정하게 상승하며, 이는 모델이 실제 객체를 놓치지 않고 잘 감지하고 있음을 나타낸다.
3. **mAP@0.5** (주황색 점선): 상당히 급격한 상승을 보이며, 모델의 전반적인 정확도와 신뢰성이 향상되고 있음을 보여준다. mAP@0.5는 모델이 객체를 얼마나 잘 감지하는지를 평가하는 중요한 지표로, 0.5 이상의 IoU(교차-연합 비율)를 가진 예측만을 고려한다.

**결론 :**

Precision, Recall, mAP@0.5와 같은 중요한 성능 지표들이 모두 상승 추세를 보이는 것으로 나타나, 효과적인 학습이 이루어지고 있다. 모델은 에포크를 거듭할수록 정확하게 객체를 감지하고 분류하는 능력이 향상되고 있다고 할 수 있다.

1. **하이퍼파라미터 튜닝 과정 및 결과 설명**

#1\_Epochs 증가: (exp2에저장)

  #Epochs: 20

#Batch size: 16

    #Learning rate: 기본 설정 사용 (0.01)

!python train.py --img 640 --batch 16 --epochs 20 --data dataset.yaml --weights yolov5s.pt

#2\_Batch size 변경: (exp3에저장)

    Epochs: 5

    Batch size: 32

    Learning rate: 기본 설정 사용 (0.01)

!python train.py --img 640 --batch 32 --epochs 5 --data dataset.yaml --weights yolov5s.pt

#3\_Learning rate 감소: (exp4에저장)

    Epochs: 5

    Batch size: 16

    Learning rate: 0.001

!python train.py --img 640 --batch 16 --epochs 5 --data dataset.yaml --weights yolov5s.pt --hyp yolov5/hyp.scratch-low.yaml

#4\_여러 파라미터 동시 조정: (exp5에저장)

    여러 하이퍼파라미터를 동시에 조정하여 최적의 조합을 찾는다.

    Epochs: 20

    Batch size: 32

    Learning rate: 0.005

!python train.py --img 640 --batch 32 --epochs 20 --data dataset.yaml --weights yolov5s.pt --hyp yolov5/hyp.scratch-med.yaml

라인, 스크린샷, 그래프, 텍스트이(가) 표시된 사진

자동 생성된 설명**그래프 분석**

라인, 스크린샷, 그래프, 텍스트이(가) 표시된 사진

자동 생성된 설명

#1\_Epochs 증가(왼쪽) vs 기본모델(오른쪽)

1. 에폭을 증가시킨 모델(두 번째 그래프)은 훈련 과정에서 더 안정적인 손실 감소와 지표 개선을 보여주며, 성능 지표들이 더욱 안정화된 것으로 보인다.

2. 더 긴 훈련은 모델이 다양한 데이터 패턴을 학습하고, 초기의 급격한 변화 이후에도 성능을 유지하거나 개선하는데 도움이 된다. 또한, mAP@0.5의 값이 긴 에폭 훈련에서 더 높게 유지되는 것으로 보아, 더 나은 일반화 성능을 보일 가능성이 높다.

결론 -> 에폭을 증가시킨 모델이 초기 빠른 학습뿐만 아니라 장기적인 성능 유지 및 안정성 면에서 더 우수한 성능을 보일 것으로 판단 할 수 있다.

텍스트, 라인, 스크린샷, 그래프이(가) 표시된 사진

자동 생성된 설명라인, 스크린샷, 그래프, 텍스트이(가) 표시된 사진

자동 생성된 설명

#2\_Batch size 변경(왼쪽) vs 기본모델(오른쪽)

1. 배치 크기를 변경한 첫 번째 그래프에서 손실 값들이 더 빠르게 감소하며, 특히 Box Loss의 감소가 두드러진다.

2. Precision, Recall, mAP@0.5 모두 첫 번째 그래프에서 더 높은 값에 도달하며, 성능 개선이 더 빠르게 일어난다.

결론 -> 배치 크기를 변경한 모델(왼쪽 그래프)은 훈련 과정에서 더 빠른 손실 감소와 더욱 개선된 성능 지표를 보여준다. 이는 배치 크기의 증가가 모델의 안정성과 학습 효율을 개선시켜 더 나은 성능을 얻는 데 기여했다고 볼 수 있다.

텍스트, 스크린샷, 라인, 그래프이(가) 표시된 사진

자동 생성된 설명라인, 스크린샷, 그래프, 텍스트이(가) 표시된 사진

자동 생성된 설명

#3\_Learning rate 감소(왼쪽) vs 기본모델(오른쪽)

1. 학습률 감소 모델은 학습 초기에 더 빠르고 급격한 성능 향상을 보여주며, 특히 mAP@0.5에서 높은 성능을 달성한다. 이는 학습률 감소가 모델이 더 빨리 최적화 포인트에 접근하도록 도와주기 때문이다.

2. 기본 모델은 비교적 일정한 학습률을 유지하면서도 좋은 성능을 보여줍니다만, 학습률 감소 모델에 비해 성능 지표가 약간 낮은 경향을 보인다.

결론 -> 학습률을 점차 감소시키는 전략은 훈련 초기에 모델이 빠르게 학습되도록 돕고, 나중에는 보다 정교한 최적화를 가능하게 하여 성능을 개선하는데 효과적이다. 이러한 접근은 특히 mAP@0.5 같은 성능 지표에서 높은 값을 얻는 데 도움이 되는거 같다.

텍스트, 도표, 그래프, 라인이(가) 표시된 사진

자동 생성된 설명라인, 스크린샷, 그래프, 텍스트이(가) 표시된 사진

자동 생성된 설명

#4\_여러 파라미터 동시 조정에 최적의 조합(왼쪽) vs 기본모델(오른쪽)

**기본 모델 (오른쪽 차트)**

* **손실 지표**: 상자 손실(Box Loss), 객체 손실(Object Loss), 클래스 손실(Class Loss)이 모두 초기에 급격히 감소하며, 이후 점차 안정화된다.
* **성능 지표**: 정밀도(Precision)와 재현율(Recall)은 비교적 일정하게 향상되며, mAP@0.5도 초기에 상승 후 안정화된다.

**파라미터 조정 모델 (왼쪽)**

* **손실 지표**: 이 모델은 훨씬 더 급격한 초기 손실 감소를 보이며, 특히 초기 에포크에서의 감소가 두드러진다. 이는 모델이 더 빨리 최적화 포인트에 도달하고 있음을 의미한다.
* **성능 지표**: 정밀도, 재현율, mAP@0.5 모두에서 더 빠른 향상을 보이며, 특히 mAP@0.5에서 상당히 높은 수치를 달성한다.

**결론**

* 파라미터 조정 모델은 기본 모델에 비해 초기 학습에서 훨씬 빠르고 효율적인 손실 감소와 성능 향상을 보여준다. 이는 파라미터 조정이 모델의 학습 효율성과 전반적인 성능을 대폭 향상시키는 데 기여하고 있음을 나타낸다.
* 특히 mAP@0.5 같은 중요 성능 지표에서 높은 값을 달성하는 것은 파라미터 조정이 모델의 예측 정확도와 관련성을 높이는 데 효과적임을 보여준다.

1. 실시간 객체 인식 결과 화면 캡처 및 분석

사람, 인간의 얼굴, 의류, 실내이(가) 표시된 사진

자동 생성된 설명텍스트, 의류, 사람, 슈트이(가) 표시된 사진

자동 생성된 설명

**첫 번째 이미지**:

* 어린이와 여성 두 명이 등장하며, 어린이는 마스크를 불완전하게 착용하고 있어 0.78의 점수를 받았다. 반면, 여성은 마스크를 올바르게 착용해 0.85의 점수를 받았다.

**두 번째 이미지**:

* 여러 남성이 마스크를 착용하지 않고 있으며, 이들의 마스크 착용 점수는 모두 0.75 이하이다.

결론 -> 이 데이터는 인공지능 모델이 다양한 환경에서 얼굴과 마스크를 어떻게 인식하고 평가하는지를 보여준다. 모델은 착용의 정확도에 따라 점수를 할당함으로써, 마스크 착용의 정확성을 자동으로 평가할 수 있는 능력을 갖추고 있음을 입증한다. 이러한 기술은 공공 보건의 관점에서 특히 중요하며, 마스크 착용 규정 준수 여부를 모니터링 하는데 유용할 수 있다. 이 분석은 해당 기술이 실제 환경에서 어떻게 작동할 수 있는지에 대한 통찰력을 제공한다.

1. 오류 사례 분석 및 개선 방안 제시

# 오류사례 분석

import pandas as pd

# 실제 라벨과 예측 라벨

ground\_truth = [0, 1, 2, 0, 1, 0, 2, 1, 1, 2]

predictions = [0, 1, 2, 0, 2, 0, 1, 1, 1, 2]

# 오류 사례 분석

error\_indices = [i for i, (gt, pred) in enumerate(zip(ground\_truth, predictions)) if gt != pred]

error\_analysis = pd.DataFrame({

    'Index': error\_indices,

    'Ground Truth': np.array(ground\_truth)[error\_indices],

    'Prediction': np.array(predictions)[error\_indices],

    'Actual Label': [f"마스크 미 착용" if x==1 else "마스크 잘 못 착용" for x in np.array(ground\_truth)[error\_indices]],

    'Predicted Label': [f"마스크 미 착용" if x==1 else "마스크 잘 못 착용" for x in np.array(predictions)[error\_indices]]

})

print("Error Case Analysis:")

print(error\_analysis)

결과 :

Error Case Analysis:

Index Ground Truth Prediction Actual Label Predicted Label

0 4 1 2 마스크 미 착용 마스크 잘 못 착용

1 6 2 1 마스크 잘 못 착용 마스크 미 착용

해석 :

Error Case Analysis

    오류 분석은 모델이 잘못 예측한 개별 사례를 보여준다.

        Index 4: 실제로는 '마스크 미 착용'(1) 상태이지만 '마스크 잘 못 착용'(2)으로 잘못 예측했다.

        Index 6: 실제로는 '마스크 잘 못 착용'(2) 상태이지만 '마스크 미 착용'(1)으로 잘못 예측했다.

개선방안 :

3.3\_시스템의 성능을 향상시킬 수 있는 방안

-가변적인 마스크 유형과 색상

마스크 유형과 색상의 다양성은 마스크 착용 상태 검출 시스템에 주요한 도전 과제이다. 시장에는 다양한 재질, 색상 및 디자인의 마스크가 존재하며, 각각의 마스크는 빛 반사, 패턴, 밀착도 등에서 차이를 보안다. 예를 들어, 광택이 나는 재질의 마스크는 강한 조명 하에서 반사되어 실제 색상이나 형태를 인식하기 어렵게 만들 수 있다. 마찬가지로, 어두운 색상의 마스크는 낮은 조명 조건에서 감지가 어려울 수 있다. 또한, 일부 패션 마스크는 기하학적 무늬나 복잡한 패턴을 포함하고 있어, 마스크의 경계를 정확히 식별하기 어려울 수 있다.

선택 이유 :

선택이유 : 이러한 문제를 극복하기 위해 데이터 다양성을 확보하고 적절한 이미지 처리 기술을 적용하는 것이 필요하다고 생각한다. 데이터셋에 다양한 유형과 색상의 마스크를 포함시키는 것은 모델이 더 넓은 범위의 시나리오에서 마스크를 정확하게 인식하도록 훈련하는 데 도움이 될 것이다. 이는 모델이 실제 세계의 복잡성을 잘 반영할 수 있게 하여 일반화 능력을 향상시킨다. 또한, 고급 이미지 전처리 기법을 통해 입력 이미지에서 색상의 영향을 줄이고 마스크의 형태와 구조에 초점을 맞출 수 있습니다. 예를 들어, 색상 정규화나 채널 분리 기법을 적용하여 모델이 주로 마스크의 형태와 텍스처를 기반으로 판단하도록 할 수 있다. 이러한 접근 방식은 특히 색상이나 패턴이 다양할 때 시스템의 정확도를 크게 향상시킬 수 있다.

데이터의 포괄적인 다양성 확보와 첨단 이미지 처리 기술의 적용은 마스크 착용 상태 검출 시스템의 성능을 극대화하고, 실제 환경에서의 유용성을 보장하는 중요한 전략이 될 것이다. 이러한 전략은 시스템이 다양한 환경과 조건에서 일관된 성능을 발휘하도록 지원함으로써, 실제 사용 시의 신뢰도와 효율성을 높이는 데 기여한다.

1. 데이터 확장과 다양화

실시: 전 세계 다양한 지역에서 마스크를 착용한 이미지를 수집하고, 특히 희귀하거나 독특한 디자인의 마스크가 포함된 데이터를 확보한다.

방법: 온라인 플랫폼과 소셜 미디어에서 이미지를 크롤링하거나, 공개 데이터셋을 활용하고, 필요에 따라 현장에서 직접 사진 촬영을 수행한다.

목적: 모델이 다양한 배경, 조명, 인종, 마스크 스타일에 대해 강인하게 작동할 수 있도록 한다.

2. 색상 정규화와 채널 분석

실시: 이미지의 색상 채널을 분석하고, 다양한 색상 스페이스(RGB, HSV, LAB 등)에서의 성능을 비교 분석한다.

방법: 색상 정규화 기술을 적용하여 이미지의 다양한 색상 대비를 줄이고, 모델이 색상보다는 형태와 패턴에 더 집중할 수 있도록 한다.

목적: 마스크의 색상이 아니라 형태와 착용 방식을 중점적으로 인식하여 성능을 향상시킨다.

3. 인공 데이터 생성

실시: GAN(Generative Adversarial Networks)을 이용해 실존하지 않는 마스크 이미지를 생성한다.

방법: 기존의 마스크 이미지를 바탕으로 새로운 유형과 색상의 마스크 이미지를 생성하여 학습 데이터셋에 추가한다.

목적: 훈련 데이터의 다양성을 인위적으로 확장하여 모델의 일반화 능력을 강화한다.

4. 실시간 피드백과 지속적 학습

실시: 사용자로부터의 피드백을 실시간으로 수집하고, 이를 모델 학습에 반영한다.

방법: 사용자 인터페이스를 통해 오류 보고를 용이하게 하고, 이 데이터를 기반으로 모델을 주기적으로 업데이트한다.

목적: 실제 사용 환경에서 발생하는 문제를 신속하게 해결하고, 모델을 지속적으로 개선하여 사용자 경험을 최적화한다.