**자율주행을 위한 객체인식 알고리즘 구현**

Team. C(아무래도): 김민성, 손관우, 오준서, 이재욱, 이하윤

**1. 서론**

**1.1 연구 배경 및 목적**

최근 인공지능 기술의 급격한 발전으로 객체 탐지(Object Detection) 분야는 다양한 응용 가능성을 제공하며 주목받고 있다. 특히 YOLO(You Only Look Once) 모델은 실시간 탐지 성능과 높은 정확도로 인해 여러 산업 분야에서 활용되고 있다. YOLO v5는 이전 버전에 비해 경량화와 처리 속도 향상을 이루어 다양한 환경에서 더욱 효과적으로 사용 가능하다.

그러나 YOLO 모델의 성능은 데이터의 질과 양, 하이퍼파라미터 설정, 학습률 스케줄링 등의 요소에 크게 의존한다. 따라서 본 연구의 목적은 데이터셋 품질을 개선하고 데이터 증강(Data Augmentation), 전이 학습(Transfer Learning), 하이퍼파라미터 튜닝 등의 최적화 기법을 체계적으로 적용하여 YOLO v5 모델의 객체 탐지 성능을 극대화하는 데 있다. 이를 통해 실무에서 활용 가능한 높은 신뢰도의 모델을 개발하고자 한다.

**1.2 연구 문제 및 중요성**

객체 탐지 기술은 다양한 응용 분야에서 핵심적 역할을 하지만, 다음과 같은 주요 문제점들이 존재한다:

• **데이터셋 부족 및 품질 저하:** 객체 탐지 모델은 대규모 데이터셋과 정확한 라벨링을 필요로 하지만, 적절한 데이터를 확보하는 것이 쉽지 않다. 잘못된 라벨링이나 데이터 품질 저하는 모델 성능에 악영향을 미칠 수 있다.

• **학습 자원의 제약:** YOLO v5와 같은 모델은 대규모 데이터 학습 시 높은 컴퓨팅 자원을 요구하며, 비효율적인 자원 활용은 학습 시간 연장과 성능 저하로 이어질 수 있다.

• **하이퍼파라미터 조정의 복잡성:** 학습률, 배치 크기, 학습률 스케줄링 등의 하이퍼파라미터 설정은 모델 성능에 중요한 영향을 미치지만, 최적의 값을 찾는 과정은 복잡하며, 경험적인 과정을 통해 최적의 값을 찾아야 한다.

<

**1.3 연구 방법 및 과정**

본 연구는 YOLO v5 모델의 성능 최적화를 목표로 다음과 같은 방법론과 범위를 포함한다:

1. **데이터셋 개선 및 증강 기법 적용:**

• 데이터셋 수집 및 정제: 다양한 객체와 환경을 포함한 고품질 데이터셋을 확보한다.

• 증강 기법 활용: 밝기, 대비, 색조 조정, 모션 블러(Motion Blur), 가우시안 노이즈(Gaussian Noise), 아핀 변환(Affine Transformation) 등 다양한 증강 기법을 적용하여 데이터 다양성을 높인다.

• 데이터 품질 관리: 과도하게 변형된 데이터는 삭제하여 데이터셋 품질을 유지한다.

2. **라벨링 품질 관리:**

• labelImg와 같은 라벨링 도구를 사용하여 정확하고 일관된 라벨링을 진행한다.

3. **학습 환경 구축:**

• 로컬 GPU 환경(CUDA)과 Google Colab 같은 클라우드 자원을 병행하여 효율적인 학습 환경을 조성한다.

4. **최적화 기법 적용:**

• 전이 학습과 Fine-tuning을 통해 초기 학습을 가속화하고, 하이퍼파라미터 최적화를 통해 학습 안정성과 성능을 개선한다.

5. **성능 평가 및 피드백:**

모델의 성능을 평가하기 위해 **mAP (mean Average Precision)**, **Precision**, **Recall**과 같은 성능 지표를 활용하였다. 이러한 지표를 통해 모델의 전반적인 탐지 능력과 정확성을 분석하고, 성능 결과를 기반으로 피드백 루프를 구축하여 지속적으로 성능을 개선하였다. 평가 결과를 바탕으로 데이터셋 보완, 하이퍼파라미터 조정, 추가 학습을 반복하여 최적화 과정을 진행하였다.

본 연구의 범위는 위 와 같은 데이터 처리, 학습 환경 설정, 최적화 기법 적용 및 성능 평가 과정을 진행하였으며, 이러한 접근법을 통해 YOLO v5 모델의 성능을 극대화하고자 한다.

**2. 이론적 배경**

* 1. **주요 개념 정의**

**- 1. YOLO v5**

YOLO v5는 CNN(Convolutional Neural Network)을 기반으로 한 객체 탐지 모델로, 입력 이미지를 단번에 처리하여 실시간 성능을 제공한다. 경량화된 구조 덕분에 빠른 학습 및 추론이 가능하다.

**- 2. 전이 학습(Transfer Learning)**

기존에 학습된 모델의 가중치를 새로운 데이터셋에 적용하여 학습 속도를 높이고, 소량의 데이터로도 높은 성능을 얻을 수 있는 기법이다.

**- 3. 데이터 증강(Data Augmentation)**

데이터셋의 다양성을 증가시키기 위해 데이터를 변형하는 기법으로, 본 연구에서는 밝기, 대비, 색조 조정, 모션 블러, 가우시안 노이즈, 아핀 변환 등 다양한 기법을 적용하였다. 또한, 과도하게 손상된 데이터는 삭제하여 데이터 품질을 유지했다.

**- 4. 평가 지표**

**1) Recall:** 모델이 실제로 존재하는 객체 중 얼마나 많은 객체를 탐지했는지를 나타내며, 탐지 누락을 최소화하는 데 중점을 둔다.

**2) Precision:** 모델이 탐지한 객체 중 올바르게 탐지한 객체의 비율을 측정하며, 낮은 오탐률을 목표로 한다.

**3) mAP (mean Average Precision):** 모델의 전반적인 탐지 성능을 평가하는 주요 지표로, 모델이 다양한 클래스에서 얼마나 잘 탐지하는지를 나타낸다.

**2.2 이론적 틀**

본 연구의 이론적 틀은 다음 세 가지 축을 중심으로 한다:

**-1. 데이터의 질과 다양성**

데이터셋의 질과 다양성은 모델의 성능에 직접적인 영향을 미친다. 본 연구에서는 다양한 환경과 객체를 포함한 데이터를 수집하고, 데이터 증강 기법을 통해 데이터의 다양성을 증가시켰다.

**-2. 전이 학습과 Fine-tuning**

Pre-trained YOLO v5 모델을 활용하여 전이 학습과 Fine-tuning을 수행함으로써 학습 시간을 단축하고 특정 도메인에 적합한 성능을 확보하였다.

**-3. 최적화 기법**

학습률 조정, 배치 크기 설정, 학습률 스케줄링 등의 하이퍼 파라미터 최적화를 통해 모델의 학습 안정성과 성능 향상을 도모하였다. 반복 실험과 피드백 루프를 통해 학습 환경을 지속적으로 개선하였다.

이 이론적 틀은 YOLO v5 모델의 성능 최적화를 위한 연구 설계와 실행의 기초가 된다.

**3. 연구 과정**

**3.1 데이터 수집 및 전처리**

- 1. 1차 데이터 수집

|  |  |  |
| --- | --- | --- |
| Class 번호 | Class 명 | 이미지 수량 |
| 0 | red | 543 |
| 1 | yellow | 1194 |
| 2 | green | 523 |
| 3 | stop\_sign | 431 |
| 4 | 40\_sign | 335 |
| 5 | 80\_sign | 447 |
| 6 | start\_sign | 20 |
| 7 | person | 772 |

Roboflow와 imgenet, ai hub에서 각 클래스에 해당하는 데이터 수집 진행하였다.

- 2. 1차 데이터 전처리

수집한 데이터들의 class 좌표가 기존에 사용할 데이터들과 호환되지 않아 전처리 작업 수행하였다.[[1]](#footnote-1)

<

- 3. 2차 데이터 수집

|  |  |  |
| --- | --- | --- |
| Class 번호 | Class 명 | 이미지 수량 |
| 0 | red | 759 |
| 1 | yellow | 636 |
| 2 | green | 772 |
| 3 | stop\_sign | 431 |
| 4 | 40\_sign | 335 |
| 5 | 80\_sign | 447 |
| 6 | start\_sign | 20 |
| 7 | person | 1228 |

기존 1차 데이터 수집 과정에서 모델의 결과를 바탕으로 red, yellow, green 데이터가 부족하다고 판단하여 해당 clas의 데이터 추가수집을 진행하였다.

- 4. 2차 데이터 전처리

수집한 데이터들의 class 좌표와 기존에 수집한 데이터들의 class 좌표 호환성을 위해 전처리 작업 수행[[2]](#footnote-2)[[3]](#footnote-3).

**3.2 데이터 어그멘테이션**

기존 1차 데이터 수집을 바탕으로 데이터의 양과 질의 부족을 확인할 수 있었고, 2차 데이터 수집 후 어그멘테이션을 진행하였다[[4]](#footnote-4). 항목은 아래와 같다.

- **밝기 조정** - **모션 블러**

- **대비조정** - **가우시안 노이즈 추가**

- **색조 변환** - **아핀 변환**

또한 위 작업들을 선택 사용하여 복합 증강 데이터를 생성하였다.

최종적으로 데이터 량은 다음과 같다

|  |  |  |
| --- | --- | --- |
| Class 번호 | Class 명 | 이미지 수량 |
| 0 | red | 1968 |
| 1 | yellow | 2059 |
| 2 | green | 2006 |
| 3 | stop\_sign | 1318 |
| 4 | 40\_sign | 1061 |
| 5 | 80\_sign | 1325 |
| 6 | start\_sign | 70 |
| 7 | person | 3851 |

**3.3 데이터 학습**

**3.1 1차 데이터 학습**

Model은 Yolo 모델중 5s를 사용하였다. 타 모델에 비해 학습 속도 및 계산양이 현재의 상황에 적합하며, 반응속도가 빨라 해당 모델을 선택하였다.

데이터는 Train, Valid, Test 항목을 **90:5:5**의 비율로 나누어 진행하였다. Epoch는 25, learning rate는 **0.01**. optimizer는 **AdamW**를 사용하였다. AdamW를 사용한 이유는 Weight Decay를 보다 정확하게 적용하여 과적합을 방지하고 일반화 성능을 향상시키기 위함이다.

**3.1 2차 데이터 학습**

1차 학습 모델과 동일하게 사용하였다.

데이터양이 증가하였기 때문에 **80:10:10**으로 데이터를 나누어 학습을 진행하였다. Epoch는 **50**으로 기존 시행의 2배수로 시행하였다. 그 외 파라미터는 동일하였다.

**4. 연구 결과**

본 연구는 YOLO v5 모델을 활용하여 객체 탐지 성능을 최적화하기 위해 **1차 데이터 학습**과 **2차 데이터 학습**을 수행하였다. 주요 실험 결과는 다음과 같으며, 분석을 위한 테스트 결과 메트릭스를 첨부하였다[[5]](#footnote-5)[[6]](#footnote-6).

**4.1 주요 결과 요약**

- 1차 결과

- **데이터 비율**: Train 90%, Valid 5%, Test 5%

- **학습 조건**: Epoch 25, Learning Rate 0.01, Optimizer: AdamW

- **평가 지표**: mAP@0.5 = **0.937**, Precision과 Recall에서 person, stop\_sign 클래스의 성능이 상대적으로 높게 나타났으나, green, red 클래스에서 일부 혼동이 발생하였다.

,

- 2차 결과

- **데이터 비율**: Train 80%, Valid 10%, Test 10%

- **학습 조건**: Epoch 50 (1차 학습 대비 2배), Learning Rate 0.01, Optimizer: AdamW

- **평가 지표**: mAP@0.5 = **0.961**, F1 점수와 Precision에서 전반적인 성능 개선이 확인되었으며, 특히 person 클래스의 성능이 88% → 91%로 향상되었다.

* **Precision**: 대부분의 클래스에서 성능이 안정화되었음.
* **Recall**: 다양한 환경에서의 인식 확률이 증가하였으며, 탐지 누락 비율이 감소하는 경향을 보였다.

**,**

* 1. **결과 해석 및 논의**

1. **모델 성능 향상:**

2차 학습에서는 데이터 양이 증가하고 Epoch를 확장함으로써 모델의 **일반화 성능**이 개선되었으며, mAP 및 F1 점수에서도 성능 향상이 확인되었다.

- **Precision**의 향상은 모델이 탐지한 객체의 신뢰도가 높아졌음을 의미하며, 이는 데이터 품질 개선 및 하이퍼파라미터 최적화의 효과로 볼 수 있다.

1. **클래스별 성능 분석**

**person 클래스**의 성능은 91%로 가장 큰 개선이 이루어졌으며, 이는 데이터 증강과 라벨링 품질 개선의 결과로 판단된다.

-다만 **green**, **red** 클래스는 여전히 다른 클래스와의 혼동 비율이 상대적으로 높게 나타났다. 이는 유사한 색상 및 시각적 특징으로 인해 모델이 분류에 어려움을 겪었을 가능성이 있다.

**3. Recall과 인식 확률**

- Recall에서 소폭 하락이 관찰된 클래스가 있으나, 이는 다양한 데이터 조건에서의 인식 확률을 높이기 위한 과정에서 발생한 현상으로 해석된다.

- 모델은 특정 클래스에 대한 예측 신뢰도를 유지하면서도 다양한 조건과 환경에서 균형 잡힌 성능을 발휘하기 위한 학습을 진행하였다.

**4. 데이터 증강 효과**

- 밝기 조정, 대비 조정, 가우시안 노이즈, 아핀 변환 등의 데이터 증강 기법을 통해 데이터 다양성이 크게 증가하였고, 이는 모델의 **일반화 능력** 향상에 기여하였다.

**5. 결론**

**5. 결론**

**5.1 연구 요약**

본 연구에서는 YOLO v5 모델의 객체 탐지 성능을 극대화하기 위해 **데이터셋 개선**, **데이터 증강**, **전이 학습** 및 **하이퍼파라미터 최적화** 기법을 적용하였다.

1. **1차 데이터 학습**에서는 **mAP@0.5 = 0.937**로 초기 모델의 성능을 확인하였다.
2. **2차 데이터 학습**에서는 데이터 양을 증가시키고 학습 Epoch를 2배로 확장함으로써 **mAP@0.5 = 0.961**로 성능이 개선되었으며, F1 점수와 Precision 또한 향상되었다.

특히, person 클래스에서의 정확도가 **88% → 91%**로 크게 개선되었으며, 다양한 환경에서도 안정적인 탐지 능력을 확보하였다.

**5.2 연구의 의의**

본 연구는 자율주행과 같은 실시간 객체 탐지가 필요한 환경에서 높은 신뢰도의 모델을 구현하기 위해 다음과 같은 의의를 가진다:

1. **데이터셋 개선과 증강**: 다양한 환경과 객체를 포함한 고품질 데이터셋 확보 및 증강을 통해 모델의 일반화 성능을 향상시켰다.
2. **전이 학습과 최적화 기법**: 기존 Pre-trained 모델을 활용하고 하이퍼파라미터 최적화를 적용하여 학습 시간을 단축하면서도 높은 성능을 확보하였다.
3. **객체 인식 신뢰도 개선**: 모델이 다양한 데이터 조건에서 균형 잡힌 성능을 발휘할 수 있도록 구현하였다.
   1. **한계점 및 향후 연구 방향**
4. **한계점**:

- red, green 클래스의 성능이 상대적으로 낮아 유사한 시각적 특징을 구분하는 데 어려움이 있었다.

- **배경과 객체** 간 혼동이 일부 존재하여 탐지 성능에 영향을 미쳤다.

- **글자 인식**에 대한 처리 과정부족 이 있다. CNN을 통한 글자 인식과 결합하여 진행하였다면 좀더 높은 인식률을 보여줄 것으로 기대된다.

1. **향후 연구 방향**:

**- 데이터셋 추가 확보**: 다양한 조건과 상황에서의 데이터를 추가 수집하여 클래스 간 불균형을 완화할 필요가 있다.

**- 하드 마이닝 기법 도입**: 오탐지 및 미탐지 사례를 분석하고 집중 학습을 통해 모델의 성능을 개선한다.

**- Threshold 최적화**: Precision과 Recall 간의 균형을 조정하여 특정 환경에 최적화된 탐지 성능을 확보한다.

* + **CNN과의 결합:** 글자 인식을 통해 비슷한 이미지들(40\_sign, 80\_sign)의 객체 탐지 성능을 개선한다.

**6. 출처**

- 데이터 목록

- <https://universe.roboflow.com/traffic-light-detection-qsrxn/traffic-light-oq7uj/browse?queryText=&pageSize=50&startingIndex=0&browseQuery=true>

- <https://www.kaggle.com/code/pkdarabi/traffic-signs-detection-using-yolov8/input>

- <https://www.image-net.org/index.php>

- <https://www.aihub.or.kr/>

1. project/preprocessing.ipynb [↑](#footnote-ref-1)
2. project/preprocessing\_additional\_data.ipynb [↑](#footnote-ref-2)
3. Project/preprocessing\_last.ipynb [↑](#footnote-ref-3)
4. project/augmentation.py [↑](#footnote-ref-4)
5. project/peprev\_runs/detect/val/ [↑](#footnote-ref-5)
6. project/runs/detect/val/ [↑](#footnote-ref-6)