YOLO11 기반 쇼핑 카트 상태 분류 시스템: CCTV 환경에서의 실시간 탐지 및 분류

YOLO11-Based Shopping Cart Status Classification System: Real-Time Detection and Segmentation in CCTV Environments

요약

본 연구는 CCTV 환경에서 쇼핑 카트의 상태(fully/empty/combined)를 자동으로 탐지하고 분류하는 실시간 시스템을 제안한다. 394장의 데이터셋으로 YOLO11n 기반 Baseline 모델을 학습한 결과 mAP50-95 0.822의 성능을 달성하였다. 이를 기반으로 YOLO11s-seg 모델과 고해상도 처리를 적용하여 원거리 객체 탐지 능력과 분류 정확도를 크게 향상시켰다. 최종 모델은 Baseline 대비 Precision 3.4% 향상, Recall 3.5% 향상을 달성하여 오탐과 미탐을 동시에 감소시켰으며 mask mAP50 0.979의 정밀한 세그멘테이션 성능을 제공한다. CCTV 테스트 영상에서 Baseline이 놓친 원거리 카트들을 성공적으로 탐지하여 실제 환경에서의 우수성을 입증하였다.

1. 서 론

1.1 연구 배경 및 필요성

대형 마트 주차장에서 정리되지 않은 쇼핑 카트로 인한 문제가 지속적으로 발생하고 있다. 방치된 카트는 주차 공간을 침해하고 바람이나 경사로 인해 차량 손상을 유발한다. 미국 법률 매체 FindLaw에 따르면 뉴저지의 한 Sam's Club 매장에서 방치된 카트가 바람에 밀려 차량 범퍼를 충돌하여 1,000달러 이상의 수리비가 발생한 사례가 보고되었다[1]. 보험 전문 매체 AutoInsurance.org도 카트로 인한 차량 손상 사고가 빈번하게 발생한다고 보고하였다[2]. 이러한 사고는 고객 불만과 법적 분쟁으로 이어질 수 있다.

기존의 카트 관리는 직원이 주기적으로 순회하며 수거하는 방식이어서 비효율적이다. 주차장 CCTV는 사후 녹화에 그쳐 실시간 대응이 어렵다[3]. 따라서 CCTV 영상으로 비어있는 카트를 자동 탐지하고 관리자에게 알림을 제공하는 시스템이 필요하다.

YOLO는 실시간 객체 탐지에 적합하며, YOLO11은 작은 객체 탐지 능력이 향상되었다[4]. 소매 환경에서 YOLOv8 기반 모델이 효과적으로 작동함이 입증되었고[5] 선반 제품 탐지 연구에서도 우수한 성능을 보였다[6]. 본 연구는 CCTV 환경에서 카트 상태를 자동 탐지하는 시스템을 제안한다. CCTV는 15~30도 하향각으로 촬영되므로 Multi-Scale Training과 Data Augmentation으로 Domain Gap 문제를 해결하였다.

1.2 연구 목적 및 범위

본 연구의 목적은 다음과 같다.

첫째, YOLO11 계열 모델로 카트 상태를 fully, empty, combined 총 3가지로 분류한다.

둘째, 객체 탐지에서 Instance Segmentation으로 확장하여 카트의 정확한 윤곽선 정보를 제공한다.

셋째, 1280px 고해상도 처리를 통해 원거리 카트의 탐지 정확도를 향상시킨다.

넷째, 실시간 처리 가능한 경량 모델로 실제 마트 환경에 배포 가능한 시스템을 제공한다.

다섯째, 394장의 제한된 데이터셋으로도 높은 정확도를 달성하는 학습 전략을 검증한다.

2. 데이터셋 구축

2.1 데이터 수집 및 구성

본 연구는 실제 CCTV 환경을 반영한 데이터셋을 구축하였다. 9곳의 마트에서 총 394장의 이미지를 수집하였으며 439개의 객체를 fully, empty, combined 3가지 클래스로 레이블링하였다. 표 1은 클래스별 분포를 나타낸다.

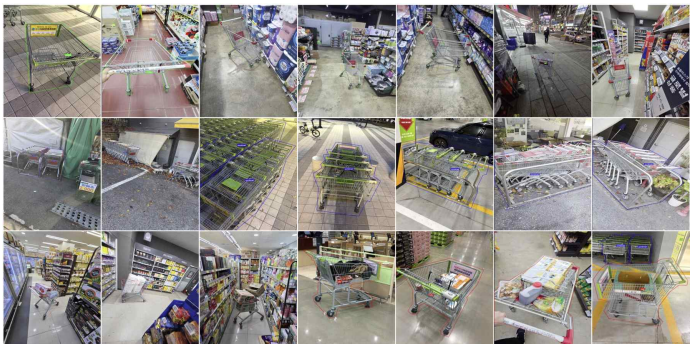
표 1. 데이터셋 클래스별 분포

클래스	객체 수	비율
fully	144	32.8%
empty	128	29.2%
combined	167	38.0%
all	439	100%

클래스별 최소 120개 이상을 확보하였으며 29.2%~38% 범위로 균형있게 구성되었다.

2.2 촬영 환경 다양화

실제 주차장 CCTV 환경을 반영하기 위해 다양한 조건에서 촬영하였다. 각도는 CCTV 하향각을 고려하여 촬영하였다. 조명은 외부주차장과 지하주차장을 모두 고려하였으며 배경은 마트 실내, 주차장, 복도 등으로 다양화하였다. 그림 1은 촬영 조건별 데이터 샘플을 보여준다.



[그림 1. 다양한 촬영 조건의 데이터 샘플]

2.3 레이블링 및 데이터 분리

Roboflow를 사용하여 Instance Segmentation 형식으로 레이블링하였으며, 2인 교차 검증으로 품질을 확보하였다. 데이터는 Train 약 80%, Validation 약 20%로 Stratified Random Split 방식으로 분리하여 클래스 비율을 유지하였다. Test 데이터는 CCTV 하향각으로 촬영하여 실제 환경 성능을 평가하였다.

3. 모델 구현 및 실험 설계

3.1 YOLO11 모델 선택

본 연구는 YOLO11 계열 모델을 선택하였다. YOLO11은 2024년 9월 출시된 최신 버전으로 쇼핑 카트의 격자 패턴과 같은 세밀한 디테일을 효과적으로 탐지할 수 있는 강점을 가지고 있다. Baseline 모델로는 가장 기본적인 YOLO11n을 선택하였다. YOLO11n은 2.6M parameters의 경량 모델로 실시간 처리가 가능하면서도 높은 정확도를 제공하여 기준 성능을 검증하기에 적합하다. 최종 모델로는 YOLO11s-seg를 선택하였다. YOLO11s는 YOLO11n 대비 더 큰 모델 용량을 가지고 있어 복잡한 패턴 인식에 유리하며 Segmentation 기능이 추가되어 카트의 정밀한 윤곽선 정보까지 제공할 수 있다.

3.2 Baseline 모델

Baseline 모델은 YOLO11n을 사용하여 객체 탐지 성능을 먼저 검증하였다. 학습 설정은 이미지 해상도 512×512, Epochs 100, Batch size 16 으로 구성하였다. 표 2는 Baseline 모델의 성능을 나타낸다. 512px 해상도에서 mAP50-95 0.822의 우수한 성능을 달성하였으며 이는 394장의 제한된 데이터셋으로도 YOLO11n의 뛰어난 성능을 보여준다.

표 2. Baseline 모델 성능

Metric	mAP50	mAP50-95	Precision	Recall
Baseline	0.962	0.822	0.938	0.926

3.3 개선 모델 설계 및 실험

Baseline의 성능을 기반으로 Instance Segmentation을 추가하여 카트의 정확한 윤곽선 정보를 제공하는 시스템을 구현하였다. 개선 전략은 다음과 같다.

(1) 모델 용량 확대: YOLO11n에서 YOLO11s-seg로 확대하여 세그멘테이션 처리 능력을 강화하였다.

(2) 이미지 해상도 증가: 512px에서 1280px로 증가시켜 원거리 카트의 미세한 격자 패턴을 보존하였다.

(3) Multi-Scale Training: 학습 중 이미지 크기를 동적으로 변경하여 다양한 거리의 카트에 대응하였다.

(4) Segmentation 최적화: overlap_mask=True로 겹치는 마스크를 처리하고 mask_ratio=4로 마스크 해상도를 조정하였으며 close_mosaic=10으로 Fine-tuning 단계에서 Mosaic 증강을 비활성화하였다.

(5) 분류 손실 가중치 조정: cls=0.5로 설정하여 Combined와 Fully 간의 오분류를 감소시켰다.

학습 설정은 이미지 해상도 1280×1280, Epochs 200, Batch size 8, AdamW optimizer (learning rate 0.001, final 0.01), Dual GPU (Kaggle T4 x2), Patience 40으로 구성하였다. 데이터 증강은 Rotation(degrees=15.0), HSV(hsv_h=0.02, hsv_s=0.8, hsv_v=0.5), Mosaic(1.0), Mixup(0.15), 기본 증강(translate=0.1, scale=0.5, fliplr=0.5)을 적용하였다. 표 3은 최종 모델의 성능을 나타낸다.

표 3. 최종 모델 (YOLO11s-seg) 성능

Metric	값
box_mAP50	0.982
box_mAP50-95	0.734
box_Precision	0.972
box_Recall	0.961
mask_mAP50	0.979
mask_mAP50-95	0.688
mask_Precision	0.972
mask_Recall	0.961

최종 모델은 Baseline 대비 Precision 3.4%, Recall 3.5% 향

상을 달성하였으며 mask_mAP50 0.979로 정밀한 세그멘테이션 성능을 제공한다.

4. 성능 평가 및 분석

4.1 정량적 성능 비교

최종 모델은 Baseline 대비 실용적 지표에서 명확한 향상을 달성하였다. mAP50은 2% 상승하였으며, Precision은 3.4%, Recall은 3.5% 향상되어 오탐과 미탐이 동시에 감소하였다. mAP50-95는 8.8% 감소하였으나, 이는 Segmentation 모델의 픽셀 단위 정밀도 요구로 인한 평가 방식의 차이에서 기인한다. 학습 난이도가 높은 Segmentation Head가 추가됨에 따라 Box Regression에 대한 최적화가 일부 분산되었다. 그러나 Precision/Recall의 동시 향상은 실제 환경에서 더 신뢰할 수 있는 탐지 성능을 의미하며 mask_mAP50 0.979를 달성하여 정밀한 윤곽선 정보까지 제공한다. CCTV 테스트 영상에서 Baseline이 놓친 원거리 카트들을 성공적으로 탐지하여 고해상도 처리의 실질적 우수성을 입증하였다. 표 4는 Baseline과 Final 모델 성능 비교를 나타낸다

표 4. Baseline vs Final 모델 성능 비교

Metric	Baseline	Final	개선율
mAP50	0.962	0.982	+2%
mAP50-95	0.822	0.734	-8.8%
mask_mAP50	-	0.979	-
Precision	0.938	0.972	+3.4%
Recall	0.926	0.961	+3.5%

4.2 실전 환경 탐지 결과

Baseline 모델은 CCTV 하향각에서 거의 탐지하지 못했으나 최종 모델은 정상 탐지하였다. 정면 환경에서는 두 모델 모두 우수한 성능을 보여 CCTV 대응력만 향상되었음을 확인하였다. 그림 2는 최종 모델의 CCTV 하향각 환경과 정면 환경에서의 탐지 결과를 보여준다.

[그림 2. Baseline vs Final 모델의 실전 탐지 결과 비교]

5. 결론 및 향후 연구

5.1 연구 성과

본 연구는 YOLO11s-seg를 활용하여 CCTV 환경에서 쇼핑 카트 상태를 자동으로 탐지하고 분류하는 실시간 시스템을 구현하였다. 객체 탐지에서 Instance Segmentation으로 확장하여 더욱 정밀한 정보를 제공할 수 있는 시스템을 개발하였다. 주요 연구 성과는 다음과 같다. 첫째, 394장의 제한된 데이터셋으로 YOLO11n Baseline 모델이 mAP50-95 0.822의 우수한 Detection 성능을 달성하였다. 둘째, YOLO11s-seg 모델과 고해상도 처리를 통해 성능 향상을 달성하였으며 Precision 3.4%와 Recall 3.5% 향상으로 오탐과 미탐이 동시에 감소하였고 원거리 객체 탐지 능력이 대폭 향상되었으며 오분류가 감소하였다. 셋째, mask_mAP50 0.979의 정밀한 세그멘테이션 성능을 달성하여 객체 탐지를 정확하게 하는 시스템을 구현하였다. 넷째, CCTV 테스트 영상에서 Baseline이 놓친 원거리 카트들을 최종 모델이 성공적으로 탐지하여 실제 환경에서의 성능 우위를 입증하였다. 기술적 측면에서는 1280px의 고해상도 처리와 Multi-Scale Training, 그리고 Segmentation 특화 튜닝이 실질적 성능 향상에 기여하였으며 Precision/Recall 지표가 mAP보다 실제 성능을 더 잘 반영함을 확인하였다.

5.2 향후 연구 방향

향후 연구 방향은 다음과 같다. CCTV 하향각 데이터를 추가 수집하여 데이터셋을 1000장 이상으로 확대하고 경계 케이스 데이터를 집중 수집하여 정확도를 개선하도록 할 계획이다.

참 고 문 헌

[1] FindLaw, "Shopping Cart Dents, Dings: Is the Store Liable?" March 21, 2019.

[2] AutoInsurance.org, "Does auto insurance cover shopping cart damage?" December 26, 2024.

[3] Coram AI, "Best Parking Lot Security Cameras and Surveillance Systems for 2025," January 10, 2025.

[4] N. Bhatti et al., "The YOLO Framework: A Comprehensive Review of Evolution, Applications, and Benchmarks in Object Detection," Computers, vol. 13, no. 12, 2024.

[5] H. Zhao et al., "Object detection in smart indoor shopping using an enhanced YOLOv8n algorithm," IET Image Processing, 2024.

[6] E. Altuntaş et al., "Object Detection in Shelf Images with YOLO," in Proc. IDAP, 2019.

Appendix: 프로그램 실행 매뉴얼

본 매뉴얼은 제안하는 모델의 학습 재현 및 성능 검증을 위한 가이드를 제공한다. 본 연구의 모든 실험 데이터와 코드는 GitHub 리포지토리에 공개되어 있으며, Google Colab 환경에서 즉시 실행 가능하다.

A. 실행 환경

본 시스템은 대용량 이미지 처리(1280px)를 위해 고성능 GPU 가속이 필수적이다. 원본 연구는 Kaggle (Tesla T4 x2) 환경에서 수행되었으나 본 매뉴얼은 독자의 접근성 및 재현 편의성을 위해 Google Colab (Tesla T4 x1) 환경을 기준으로 재구성되었다.

- **Platform:** Google Colab
- **Hardware:**
  - **GPU:** NVIDIA Tesla T4 (Single GPU)
  - **VRAM:** 12GB 이상
- **Software:**
  - **Python:** 3.10 이상
  - **Library:** Ultralytics YOLO 8.3+, PyTorch 2.0+, CUDA 11.8+

B. 리포지토리 정보

실행에 필요한 소스 코드, 데이터셋, 학습된 가중치 파일은 아래 리포지토리에서 제공된다.

URL: https://github.com/ICT-Top-Bottom/MCA-Appendix

포함 내용: 실행 코드(MCA_Appendix.ipynb), 데이터셋(394장), 학습된 가중치(best.pt), data.yaml

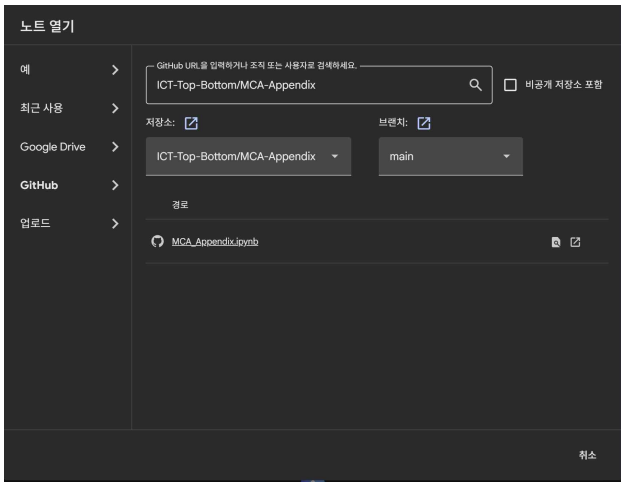
C. 실행 순서

Step 1: Google Colab 노트북 생성 및 설정

1. Google Colab(https://colab.research.google.com)에 접속하여 좌측 하단 파란색 [새 노트]를 클릭한다.
2. 상단 메뉴의 [런타임] → [런타임 유형 변경]을 클릭한다.
3. 하드웨어 가속기를 T4 GPU로 설정하고 [저장]을 누른다.

Step 2: GitHub 코드 불러오기 (Import)

1. 노트북 상단 메뉴에서 [파일] → [노트 열기]를 클릭한다.
2. [GitHub] 탭을 선택하고 검색창에 리포지토리 주소(ICT-Top-Bottom/MCA-Appendix)를 입력한다.
3. 검색 결과 목록에서 MCA_Appendix.ipynb를 클릭하여 연다.



이 과정이 완료되면 논문의 모든 실행 코드가 노트북 셀에 자동으로 로드된다.

Step 3: 실행 (Run)

1. [Cell 1] & [Cell 2] (환경 설정): 필수 라이브러리를 설치하고 GitHub에서 데이터셋을 다운로드한다. (최초 1회 실행)
주의: 세션이 만료되어 재접속할 경우 이 단계를 다시 실행해야 한다.
2. [Cell 3] (모델 학습): 제안 모델의 학습 파라미터가 적용된 학습을 시작한다.
주요 설정: imgsz=1280, batch=4, mixup=0.15

3. [Cell 4] (추론 검증): 학습 과정을 생략하고 싶다면 이 셀을 실행하여, 동봉된 best.pt 모델로 테스트 이미지에 대한 추론 결과를 즉시 확인한다.

D. 최적화 및 주의사항

메모리 부족(OOM) 대응 : imgsz=1280은 고해상도 설정이므로 메모리 부족 에러 발생 시 batch사이즈를 4에서 2로 줄이거나, imgsz를 1024 또는 640으로 하향 조정한다.

원본 성능 재현 : 논문과 동일한 성능을 완벽하게 재현하려면 Kaggle Notebook환경을 사용해야 한다. Kaggle에서 Accelerator : GPU T4 x2를 설정한 후, 코드 내 설정을 device=[0, 1], batch=8로 변경하여 학습을 수행한다.

데이터셋 경로 오류 : 데이터셋을 찾을 수 없다는 에러가 발생할 경우, [Cell 2]가 정상적으로 실행되어 좌측 파일 탐색기(/content/)에 MCA-Appendix폴더가 생성되었는지 확인한다.