

운전 미숙자를 위한 운전 보조 애플리케이션



202155516 김대길

202155536 김주송

202155592 이지수

지도교수 박진선

목 차

1. 서론	1
1.1. 연구 배경.....	1
1.2. 기존 문제점.....	2
1.3. 연구 목표.....	2
2. 연구 배경	4
2.1. AI 모델 개발.....	4
2.1.1. YOLOv8(You Only Look Once version 8)	4
2.1.2. OpenCV	5
2.2. 서비스 개발.....	7
2.2.1. 개발 언어 및 프레임워크	7
2.2.2. 개발 도구.....	8
3. 연구 내용	9
3.1. AI 모델 개발.....	9
3.1.1. 차량 인식 모델.....	9
3.1.2. 신호 인식 모델.....	9
3.1.3. 차선 인식 모델.....	10
3.2. 서비스 개발.....	12
3.2.1. 운전 보조 기능.....	12
3.2.2. 초보 운전 경로 안내 기능	13
3.2.3. 운전 습관 피드백 기능	16
3.2.4. 긴급 상황 대응 기능.....	20

4. 연구 결과 분석 및 평가.....	21
4.1. AI 모델 성능 평가.....	21
4.1.1. 차량, 사람, 신호등 탐지 (K_Traffic.pt)	21
4.1.2. 차선 탐지.....	24
4.2. 서비스 구현.....	25
4.2.1. 기능 트리.....	25
4.2.2. 플로우 차트	25
4.2.3. 유스 케이스	28
4.2.4. 와이어 프레임.....	30
5. 결론 및 향후 연구 방향.....	31
6. 구성원별 역할 및 개발 일정	33
6.1. 구성원별 역할	33
6.1.1. 김대길	33
6.1.2. 김주송	33
6.1.3. 이지수	33
6.2. 개발 일정.....	34
7. 참고 문헌	34

1. 서론

1.1. 연구 배경

운전은 현대 생활에서 필수적인 기술이다. 경찰청에서 제공한 2023년 교통통계분석에 따르면, 2022년 기준으로 전체 인구의 약 66%가 운전 면허를 소지하고 있고 이러한 통계는 운전이 개인의 이동과 일상생활에서 중요한 역할을 하고 있음을 보여준다.

2022년 운전면허소지인구비율



그림 1. 2022년 운전 면허 소지 인구 비율
(출처: 경찰청 2023년 교통사고 통계)

그러나 많은 초보 운전자들은 도로 위에서 불안과 어려움을 겪고 있다. 이들은 교통 상황에 대한 빠른 판단, 차량 조작의 숙련도, 그리고 교통 신호의 인지 등에 어려움을 느끼며, 이는 교통 사고의 위험성을 높인다. 초보 운전자들은 주로 차량 조작 미숙과 판단 오류로 인해 사고를 일으키는 경향이 있다.

운전면허 취득경과년수별 교통사고 건수

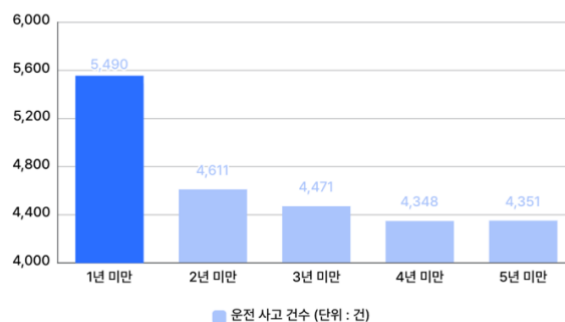


그림 2. 2022년 운전면허 취득경과년수별 교통사고 건수
(출처: 경찰청 2023년 교통사고 통계)

고령 운전자들 또한 도로 위에서 많은 어려움을 겪고 있다. 도로교통공단에서 제공한 2022년 교통사고 통계분석에 따르면, 전체 교통사고건수는 전년대비 3.1% 감소한 반면, 고령 운전자가 일으킨 사고는 전년대비 2.5% 증가한 형태를 보인다. 고령 운전자들은 반응 속도의 저하와 인지 기능의 약화로 인해 사고 위험이 높다.

운전 사고 연도별 현황



고령 운전자 사고 연도별 현황



그림 3. 운전 사고 연도별 현황과 고령 운전자 사고 연도별 현황 비교
(출처: 도로교통공단 2022년 교통사고 통계분석)

1.2. 기존 문제점

교통 사고는 개인과 사회 모두에게 큰 피해를 초래할 수 있으므로, 이를 줄이기 위한 노력이 필요하다. 기존의 내비게이션 시스템은 복잡한 상황에서는 운전 미숙자가 이해하기 어려운 표현을 사용할 수 있다는 한계가 있다. 또한, 일반적으로 카메라 기반 객체 탐지 기능이 없어 단순한 텍스트 기반의 경고 메시지만 제공하는 경우가 많다.

현재 시행되고 있는 제도인 운전 면허 반납제와 시행 고려 중인 조건부 운전 면허도 고령자들의 운전만을 제한하는 데 중점을 두고 있으며, 이는 실질적인 해결을 제공하지 못하고 있다. 초보 운전자와 고령 운전자들에게는 보다 맞춤형이고 세심한 운전 보조가 필요하다.

1.3. 연구 목표

본 프로젝트의 목표는 운전 미숙자(초보 운전자 및 고령 운전자)를 위한 맞춤형 운전 보조 애플리케이션을 개발하여, 이들의 운전 경험을 개선하고 교통 안전을 증진하는 것

이다. 기존의 일반 운전자들을 대상으로 한 플랫폼과 차별화하여, 특별한 지원이 필요한 운전 미숙자들에게 보다 세심하고 맞춤형 서비스를 제공한다. 이를 통해 고령화 사회에서 증가하는 고령 운전자 사고 문제를 해결하는 데 기여할 것이다. 구체적인 목표는 다음과 같다.

- (1) 실시간 교통 상황 파악: 스마트폰 후면 카메라와 GPS 기능을 활용하여 운전자가 실시간으로 교통 상황을 파악하고 안전하게 운전할 수 있도록 지원한다.
- (2) 안전 운전 경고 제공: 전방 차량과의 안전 거리 유지, 차선 이탈 방지, 제한 속도 준수를 위한 실시간 경고를 제공한다.
- (3) 안전한 운전 경로 안내: 초보 운전자가 연습하기 좋은 교통량이 적고 사고 발생률이 낮은 안전한 도로 구역을 안내한다.
- (4) 운전 교육 및 피드백 제공: 운전 기록을 대시보드 형태로 제공하여 운전 습관을 개선하고, 점수나 간단한 평가를 얼굴 표정으로 표현해 재미있게 연습할 수 있는 환경을 조성한다.
- (5) 사고 대응 지원: 사고 발생 시 상황별 대응법 안내와 자동차 보험 회사와의 신속한 연결을 지원한다.

이 애플리케이션을 통해 운전 미숙자들이 도로 위에서 느끼는 불안감을 해소하고, 보다 안전하고 자신감 있게 운전할 수 있도록 돕는 것을 목표로 한다. 또한, 고령 운전자 사고 증가 문제를 해결하는 실질적인 방안 중 하나로 작용할 것이다.

2. 연구 배경

2.1. AI 모델 개발

2.1.1. YOLOv8(You Only Look Once version 8)

(1) YOLO의 기본 원리

YOLO의 기본 개념은 이미지에서 객체를 탐지하기 위해 단 하나의 신경망을 사용하는 것이다. 이미지 전체를 하나의 네트워크에 통과시켜 동시에 바운딩 박스와 클래스 확률을 예측한다. 이 방법은 이미지의 여러 부분을 여러 번 처리하는 기존의 R-CNN이나 Fast R-CNN과 달리, 단일 패스 방식으로 매우 빠르게 객체 탐지가 가능하다.

YOLO 모델은 이미지를 $S \times S$ 크기의 그리드로 나누고, 각 그리드 셀에서 객체의 존재 여부를 판단한다. 각 셀은 고정된 수의 바운딩 박스와 해당 객체에 대한 신뢰도 점수를 출력하며, 이 신뢰도는 해당 셀에서 객체가 있을 확률과 그 객체의 정확도를 곱한 값이다. 이후, 예측된 바운딩 박스들 중에서 겹치는 부분을 제거하기 위해 Non-Maximum Suppression(NMS)을 사용하여 최종적으로 탐지된 객체를 결정한다.

(2) YOLOv8의 주요 특징

YOLOv8은 YOLOv4와 YOLOv5를 기반으로 성능 개선이 이루어졌으며, YOLOv6와 YOLOv7의 최적화 기법을 일부 도입하여 더 나은 성능을 제공한다. YOLOv8의 주요 개선 사항은 다음과 같다.

- 백본 네트워크 개선: YOLOv8은 CSPNet(Cross Stage Partial Network)을 활용하여 네트워크의 효율성을 높였다. 이를 통해 특성 맵의 중요 정보 손실을 줄이면서도 모델의 경량화를 달성할 수 있다.
- Head 구조 최적화: YOLOv8은 FPN(Feature Pyramid Network)과 PAN(Path Aggregation Network)의 결합 구조를 사용하여, 객체의 크기와 위치에 관계 없이 더욱 정확한 탐지를 수행할 수 있도록 하였다.
- 기타 최적화 기법: 모델 파라미터 수와 연산량을 줄여 모바일 및 임베디드 환경에서도 효율적으로 동작할 수 있도록 개선되었다. 특히, ONNX 및 TFLite로의 모델 변환이 용이해져 다양한 환경에서의 적응성을 높였다.

(3) YOLOv8을 사용한 차량 및 신호등 인식

본 연구에서는 YOLOv8을 사용하여 차량 및 신호등을 인식하는 모델을 개발하였다. AI Hub에서 제공하는 다양한 데이터셋을 활용하여 모델을 학습시켰으며, 특히 차량, 보행자, 이륜차 등의 객체와 신호등 및 도로 표지판을 인식하는 데 중점을 두었다.

- ① 데이터 전처리: 학습에 사용된 데이터는 다양한 교통 상황을 반영하여, 주간 및 야간, 맑은 날씨와 흐린 날씨 등 다양한 조건에서의 데이터를 포함하였다. 데이터는 모델에 적합한 형식인 640x640 픽셀로 리사이즈되었고, 객체의 위치 정보는 YOLO의 형식에 맞게 `<class_id>` `<x_center>` `<y_center>` 형식으로 변환되었다. 이를 통해 다양한 크기와 위치의 객체를 효과적으로 학습할 수 있도록 준비하였다.
- ② 모델 학습 및 평가: 모델은 YOLOv8의 최신 기능을 활용하여 학습되었으며, 크로스 엔트로피 손실 함수와 IoU(Intersection over Union)를 기반으로 객체 탐지 정확도를 최적화하였다. 학습된 모델은 다양한 교통 상황에서의 검증 데이터셋을 사용하여 정확도(Accuracy), 정밀도(Precision), 재현율(Recall), F1 스코어 등의 지표를 통해 평가되었다. 이를 통해 모델의 성능을 지속적으로 모니터링하고 개선할 수 있었다. 특히, 다양한 교통 상황에서의 성능을 중점적으로 평가하여 모델이 실제 환경에서도 안정적으로 작동할 수 있도록 하였다.
- ③ 실시간 모델 추론: 학습된 YOLOv8 모델은 실시간으로 차량과 신호등을 탐지하며, 높은 신뢰도의 탐지 결과만을 화면에 표시하도록 설정되었다. 이를 통해 도로 상의 차량 흐름을 모니터링하고, 신호등 상태를 정확하게 파악할 수 있었다.

2.1.2. OpenCV

OpenCV는 컴퓨터 비전과 이미지 처리 작업을 수행하기 위한 오픈 소스 라이브러리로, 다양한 영상 처리 기능을 제공하여 실시간 애플리케이션 개발에 유용하게 활용된다. 특히, 차선 인식과 같은 선형 구조 검출에서 OpenCV의 Canny Edge 검출과 Hough Line Transform 알고리즘은 매우 효과적인 도구로 사용된다.

(1) OpenCV의 기본 원리

OpenCV는 픽셀 기반의 이미지 처리 기능을 통해 다양한 변환과 필터링 작업을 수행할 수 있다. 이를 통해 객체의 형태를 분석하고, 관심 영역(ROI) 설정, edge 검출, 기하학적 변환 등을 통해 이미지에서 필요한 정보를 추출할 수 있다.

- Canny Edge 검출: Canny Edge 검출은 이미지의 강한 경계선을 감지하여 추출하는 알고리즘이다. 그레이스케일로 변환하고 Gaussian Blur를 적용하여 노이즈를 제거한 후, Canny 알고리즘을 통해 이미지 내의 에지들을 감지한다. 이를 통해 차선과 같은 선형 구조를 효과적으로 검출할 수 있다.
- Hough Line Transform: Hough 변환은 에지 이미지에서 직선을 검출하는 알고리즘으로, 선의 기울기와 절편을 파라미터 공간에서 누적하여 직선을 찾는다. 이를 통해 차선과 같은 도로 상의 직선 구조를 감지할 수 있다.

(2) 차선 인식을 위한 OpenCV의 적용

본 연구에서는 OpenCV를 사용하여 차선 인식 모델을 개발하였다. 차선 인식은 차량의 도로 유지 상태를 모니터링하고, 운전자가 차선 이탈 여부를 감지하는 데 중요한 역할을 한다.

- ROI 설정: 도로 상의 차선은 특정 영역에만 존재하기 때문에, 이미지에서 관심 영역(Region of Interest, ROI)을 설정하여 도로 부분만 남기고 나머지 부분을 제거했다. 이를 통해 불필요한 배경 정보로 인한 오탐지(false positive)를 줄이고, 차선 감지 성능을 향상시켰다.
- 차선 인식 알고리즘: Canny Edge 검출을 사용하여 도로의 경계선을 감지한 후, Hough Line Transform을 적용하여 이미지 내의 차선을 직선 형태로 검출했다. 검출된 직선들은 왼쪽 차선과 오른쪽 차선으로 분류되어, 각각 선형 회귀를 통해 연장선을 계산하고 도로의 차선 영역을 정의했다.
- 차선 영역 채색: 인식된 차선 사이의 영역을 fillPoly 함수를 사용하여 색칠함으로써, 차선 영역을 시각적으로 강조했다. 이렇게 색칠된 차선 영역은 도로의 주행 경로를 시각적으로 명확하게 표시하여, 운전자가 차선을 이탈하지 않도록 도울 수 있다.

(3) 차선 이탈 감지

차선 인식 모델은 차량의 현재 위치와 도로 상의 차선 위치를 비교하여, 차량이

도로 내에서 올바르게 주행하고 있는지 모니터링할 수 있다. 차량의 중심 위치가 차선 영역 밖으로 벗어날 경우, 차선 이탈로 판단하여 운전자에게 경고를 줄 수 있다. 이를 통해 차선 유지 보조 시스템(Lane Keeping Assist System, LKAS)의 기능을 구현할 수 있다.

OpenCV를 통한 차선 인식 기술은 YOLOv8 기반의 차량 및 신호등 인식 모델과 결합하여, 실제 도로 환경에서의 차량 위치와 차선 정보를 종합적으로 분석하고 안전한 주행을 위한 데이터 기반의 의사 결정을 가능하게 한다.

2.2. 서비스 개발

2.2.1. 개발 언어 및 프레임워크

(1) Kotlin 1.9.0

안드로이드 애플리케이션 개발을 위한 프로그래밍 언어로 사용자 인터페이스 구현 및 네트워크 통신, 데이터 관리 등의 전반적인 앱 로직을 처리한다.

(2) Tensorflow Lite 2.14.0

경량화된 머신러닝 프레임워크로 모바일 및 임베디드 장치에서 딥러닝 모델을 실행할 수 있도록 최적화된 라이브러리이다. 앱 내에서 객체 인식 및 분석 등의 AI 기능을 제공한다.

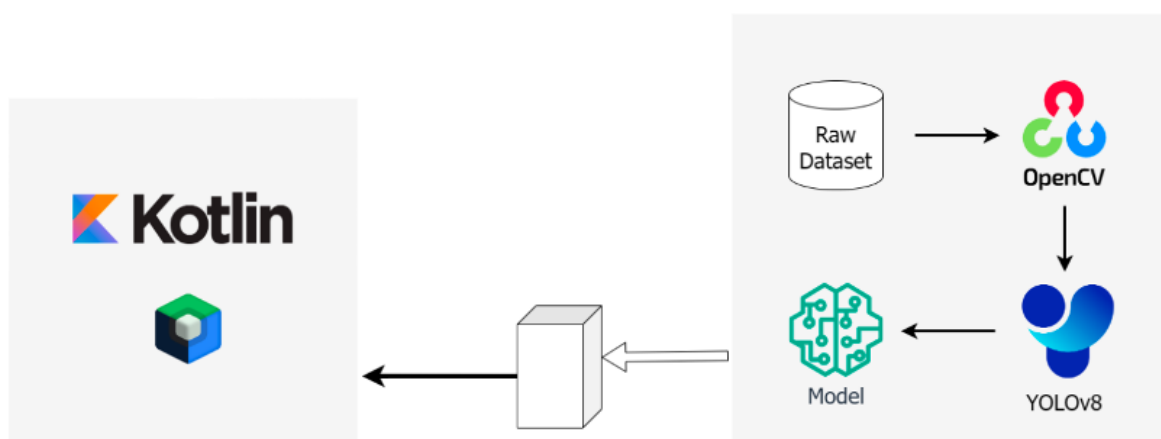


그림 4. 애플리케이션 시스템 설계도

2.2.2. 개발 도구

(1) T Map API

SK에서 제공하는 지도 API로 경로 안내 기능과 장소 검색 기능을 구현했습니다. 사용자 위치를 기반으로 가장 안전한 경로를 계산하고, 교통 사고 다발 지역을 피할 수 있도록 안내하는 핵심 기능을 제공합니다. 교통 상황 및 다양한 지도 데이터를 이용하여 사용자에게 직관적인 인터페이스를 제공하고, 실시간 데이터를 반영했습니다.

(2) Google Map API

구글에서 제공하는 지도 API로 사용자의 현재 위치와 목적지 간의 경로를 제공합니다. 전 세계적으로 신뢰할 수 있는 데이터를 제공하며, 프로젝트의 위치 기반 서비스의 정확도를 높이기 위해 함께 사용했습니다.

(3) 한국도로교통공단 Open API

한국 도로 교통 관련 데이터를 활용하여 교통 사고 다발 지역 및 위험 구역 정보를 실시간으로 표시합니다. 이를 기반으로 초보 운전자가 사고 위험 지역을 미리 인지하고 안전한 경로를 선택할 수 있도록 지원합니다

3. 연구 내용

3.1. AI 모델 개발

3.1.1. 차량 인식 모델

본 프로젝트에서는 AI Hub에서 제공하는 에이모(Aimo)가 구축한 차량 및 사람 인지 영상 데이터를 활용하여 차량 인식 모델을 개발하였다. 사용된 데이터는 '일반차량', '보행자', '목적차량(특장차)', '이륜차' 등의 다양한 객체를 포함하고 있다.

우선, 데이터 전처리 과정에서 전체 데이터셋에서 필요한 객체 데이터만을 추출하고, 불필요한 데이터를 제거하였다. 추출된 라벨링 데이터는 YOLO 모델의 입력 형식에 맞춰 "`<class_id> <x_center> <y_center> <width> <height>`"의 형식으로 변환하였다. 여기서 각 클래스는 사전에 정의된 `class_id`로 분류되었으며, 각 객체의 중심 좌표와 크기는 이미지 크기에 대한 상대 좌표로 처리되었다.

데이터는 학습 효율성을 높이기 위해 `train(70%)`과 `val(30%)` 폴더로 나누어 저장하였다. 학습 과정에서는 주로 YOLOv8 모델을 활용하였으며, 최종적으로, 모델의 성능은 test set을 통해 평가되었으며, 정확도(Accuracy), 정밀도(Precision), 재현율(Recall), F1 스코어 등의 지표를 통해 성능을 분석하였다.

3.1.2. 신호 인식 모델

신호 인식 기능 구현을 위해, 교통 신호 데이터를 AI Hub에서 라이드플렉스가 구축한 신호등/도로표지판 인지 영상(수도권) 데이터를 추가로 확보하여 YOLO 모델에 적용하였다. 주요 데이터는 신호등, 직진, 좌회전, 멈춤 신호 등 다양한 교통 신호를 포함하고 있으며, 각 신호는 고유의 클래스 id로 구분되었다.

본 신호 인식 모델에서는 특히 야간 환경 등 다양한 조건에서의 신호 인식 성능을 높이기 위해 야간에서의 데이터를 포함하였다. 데이터 전처리 과정에서 이미지의 명암비를 조정하여 모델이 다양한 조명 환경에서도 일관된 성능을 발휘할 수 있도록 했다. 또한, 신호등과 같은 작은 객체의 인식 정확도를 높이기 위해 YOLOv8의 최신 버전에서 제공하는 앵커 박스(Anchor Box) 튜닝 기법을 사용하였다.

신호 인식 모델은 차량 인식 모델과 마찬가지로 `train`과 `validation` 데이터셋으로 분리하여 학습이 이루어졌으며, 학습 과정에서는 주로 YOLOv8 모델을 활용하였고 모델의 성능

은 test set을 통해 평가되었으며 mAP50 등의 지표를 통해 분석하였다.

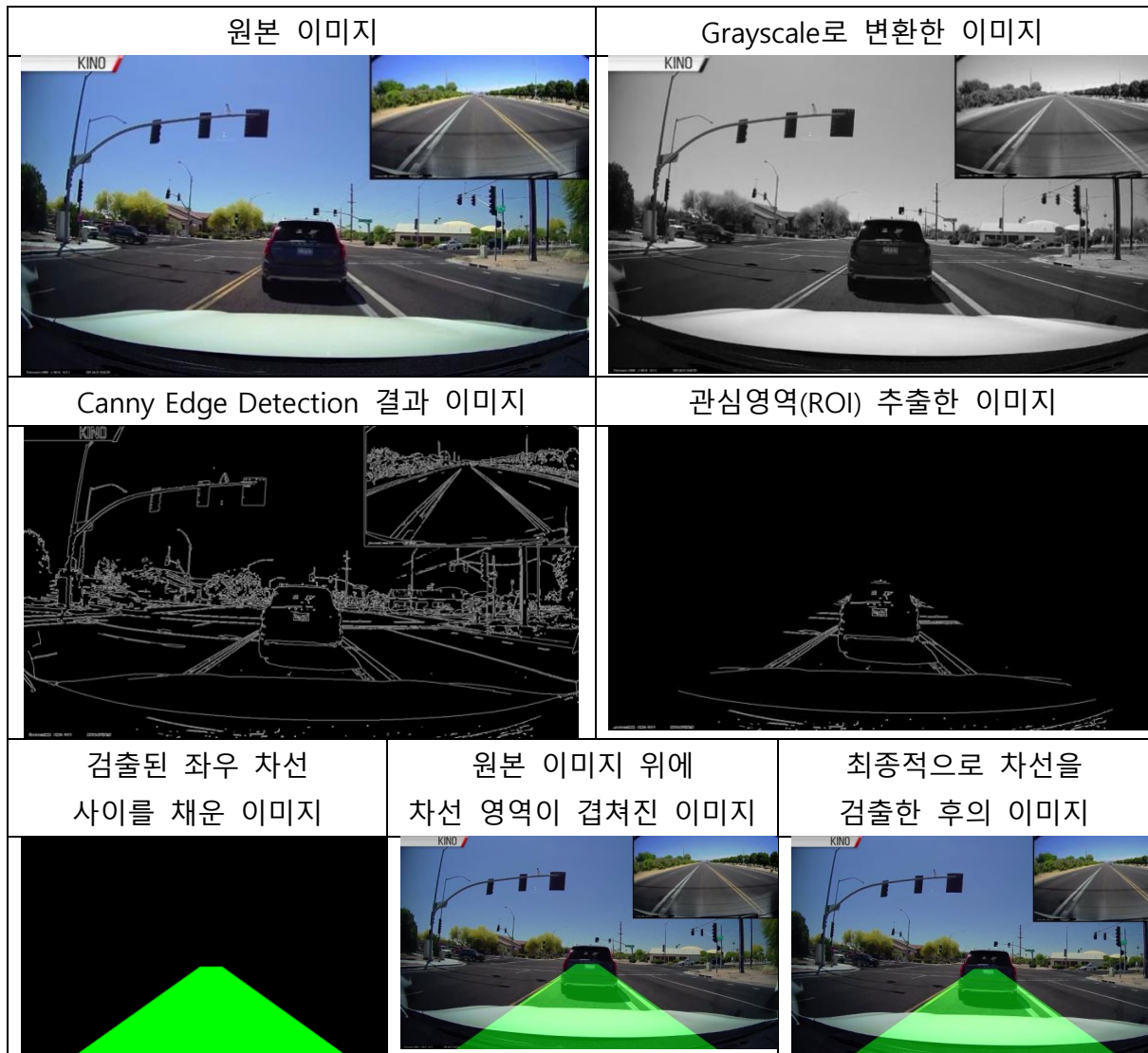
3.1.3. 차선 인식 모델

본 연구에서는 차선 인식을 위해 OpenCV를 사용하여 이미지 처리와 선형 구조 검출을 수행했다. 차선 인식은 도로에서 운전자가 차선을 이탈하지 않고 안전하게 주행할 수 있도록 돕는 중요한 기능 중 하나이다. 차량의 현재 위치와 차선 정보를 기반으로 차선 이탈 여부를 감지할 수 있는데, 본 모델에서는 Canny Edge 검출과 Hough Line Transform을 활용하여 차선 인식 과정을 진행했다. 상세한 과정은 다음과 같다.

- (1) 데이터 전처리: 입력 영상은 먼저 그레이스케일로 변환되었다. 이는 색상 정보를 제거하고 엣지 검출의 정확성을 높이기 위한 과정이다. 그레이스케일 변환 후, Canny Edge 검출 알고리즘을 사용하여 이미지에서 강한 경계선을 찾아내었다. Canny 알고리즘은 두 개의 임계값을 사용하여 픽셀 강도 변화가 급격한 부분을 선명하게 강조해 준다. 이 과정에서 발생하는 엣지들은 차선의 경계가 될 가능성이 높은 영역으로 분류된다.
- (2) 관심 영역 설정: 차량이 주행하는 도로의 차선은 특정 영역에만 존재하므로, 관심 영역(ROI, Region of Interest)을 설정하여 차선이 있을 가능성이 높은 부분만을 추출했다. 이때 사용된 ROI는 도로의 하단부터 이미지의 중앙을 향해 삼각형 형태로 설정하였으며, 이 영역만을 대상으로 엣지 검출 결과를 분석했다. ROI를 사용함으로써 불필요한 부분의 노이즈를 줄이고, 검출 성능을 높일 수 있었다.
- (3) Hough Line Transform을 통한 차선 검출: 관심 영역(ROI)에서 검출된 엣지를 바탕으로 Hough Line Transform을 적용하여 선형 구조인 차선을 검출했다. Hough Transform은 이미지에서 점들의 집합을 직선으로 변환하여 이를 찾아내는 알고리즘으로, 차선과 같은 선형 패턴을 추출하는 데 적합하다. 이 알고리즘을 통해 추출된 직선 중, 경사도를 이용해 좌우 차선을 분류하고, 직선의 방향에 따라 좌측 차선과 우측 차선을 구분했다.
- (4) 차선 보간 및 시각화: Hough Transform으로 검출된 차선들은 각도에 따라 분류되어 좌우 차선을 나타내지만, 실제로 차선이 연속적으로 이어지도록 보이기 위해 보간 과정이 필요하다. 이를 위해 각 차선의 점들을 기반으로 1차 다항식을 사용해 직선을 보간하였고, 차량의 주행을 고려하여 일정 범위에서 차선이 연결되도록 하였다. 이 과정에서 이미지 상단과 하단의 좌표를 이용해 차선을 연장

했다.

- (5) 차선 사이 영역 색칠: 검출된 좌우 차선의 좌표를 이용하여 그 사이의 영역을 색칠했다. 좌우 차선의 점을 연결하여 다각형을 형성하고, 이를 OpenCV의 fillPoly 함수로 채워넣어 차선 사이의 면적을 색칠했다. 이를 통해 운전자는 차선을 시각적으로 더욱 명확하게 볼 수 있다.
- (6) 차선 인식 성능 평가 및 개선: 최종적으로, 차선 인식의 정확도를 평가하기 위해 다양한 도로 상황(주간, 야간, 비가 오는 날씨 등)에서 테스트를 진행했다. 평가 결과를 바탕으로, Canny Edge 검출의 임계값을 조정하고, Hough Transform의 파라미터를 최적화하여 인식 성능을 개선했다. 이러한 과정을 통해 차선 인식 모델은 다양한 환경에서도 일관된 성능을 발휘할 수 있도록 개발되었다.



3.2. 서비스 개발

3.2.1. 운전 보조 기능

스마트폰의 후면 카메라를 활용하여 실시간으로 도로 상황을 분석하고, 다양한 운전 보조 기능을 제공한다. 이 기능들은 YOLOv8 객체 탐지 모델과 OpenCV를 기반으로 한 영상 처리 알고리즘을 사용하여 도로 상황을 실시간으로 분석하고 운전 보조 기능을 수행한다.

(1) 차선 인식 기능

- 차선 이탈 경고: OpenCV를 이용한 차선 감지 알고리즘으로 후면 카메라의 영상을 분석하여 현재 차량이 주행하는 차선의 경계를 검출한다. 차량의 위치가 검출된 차선의 경계를 벗어나는 경우, 운전자가 차선을 이탈했다고 판단하고 "차선을 이탈하였습니다."라는 음성 경고를 제공한다.

(2) 신호 인식 기능

- 신호 변경 안내: 후면 카메라를 통해 신호등 변화를 감지하고, YOLOv8 모델을 활용하여 신호등의 상태를 분석한다. 신호가 초록불로 변경되면 "초록불로 변경되었습니다."라는 음성 안내를 제공하여 운전자에게 알린다.
- 신호 위반 경고: 신호를 위반할 경우, "신호를 위반하였습니다. 운전 중 주의해주세요."라는 경고 안내를 통해 안전 운전을 유도한다.

(3) 차량 간 거리 계산 기능

- 안전 거리 경고: 후면 카메라로 전방 차량을 감지하고, 바운딩 박스의 y 좌표를 사용하여 감지된 차량 간의 거리를 계산한다. 이때, 차량 간의 거리가 안전 거리 임계값보다 작아지면 안전거리가 확보되지 않았다고 판단하고 "안전 거리를 확보하세요."라는 음성 안내를 제공한다.

(4) 전방 차량 출발 감지 기능

- 정체 상황 안내: 정체된 상황이었다가 전방 차량이 움직이기 시작할 때 감지된 차량의 위치 변화량을 계산한다. 이전 프레임의 차량 위치와 비교하여 설정된 임계값 이상의 변화가 감지되면, 전방 차량이 출발했다고 판단하고 "전방 차량이 출발하였습니다."라는 음성 안내를 제공한다.

3.2.2. 초보 운전 경로 안내 기능

사용자가 목적지를 입력하면 교통 사고 다발 구역 데이터를 기반으로 안전한 경로를 재 탐색한다. 이 과정에서 세 가지 주요 알고리즘이 적용되며, 각각의 알고리즘은 사고 위험을 최소화하고 효율적인 경로를 제공하는 역할을 한다.

(1) 사고 위험도 측정 알고리즘

사고 데이터(발생 횟수, 사상자 수 등)를 기반으로 특정 지역의 사고 위험도를 계산하기 위해 가중치 기반의 위험도 계산 알고리즘을 사용한다. 이 알고리즘은 사고 발생 빈도와 피해 정도를 반영하여 위험 점수를 산출한다.

① 사고 위험도 측정 알고리즘

사고 데이터(발생 횟수, 사상자 수 등)를 기반으로 특정 지역의 사고 위험도를 계산하기 위해 가중치 기반의 위험도 계산 알고리즘을 사용한다. 이 알고리즘은 사고 발생 빈도와 피해 정도를 반영하여 위험 점수를 산출한다.

입력:

사고 발생 횟수 (ocrrncCnt), 사상자 수 (casltCnt), 사망자 수 (dthDnvCnt), 중상자 수 (seDnvCnt), 경상자 수 (slDnvCnt), 부상자 수 (wndDnvCnt)

출력:

사고 위험도 점수 (severityScore)

1. 위험도 점수를 초기화한다:

$severityScore = 0$

2. 각 항목에 가중치를 곱하여 위험도 점수를 계산한다

$severityScore += (ocrrncCnt * 0.4)$

$severityScore += (casltCnt * 0.2)$

$severityScore += (dthDnvCnt * 0.2)$

$severityScore += (seDnvCnt * 0.1)$

$severityScore += (slDnvCnt * 0.05)$

$severityScore += (wndDnvCnt * 0.05)$

3. 최종 위험도 점수를 반환한다.

종료

② 교통 사고 다발 구역 확인 알고리즘

사용자가 사고 다발 구역에 진입했는지 확인하기 위해 짝수 교차 알고리즘(Even-Odd Rule)을 사용한다. 이 알고리즘은 사용자가 특정 영역(사고 다발 구역)의 경계선을 몇 번 교차했는지 계산하며, 짝수 번 교차하면 사용자가 구역 밖에 있고, 홀수 번 교차하면 구역 안에 있다고 판단한다.

입력:

사고 다발 구역의 경계 좌표 (polygon), 사용자의 현재 위치 좌표 (point)

출력:

사용자가 구역 안에 있는지 여부 (True/False)

1. 교차 횟수를 초기화한다:

crossCount = 0

2. 사고 다발 구역 경계선의 각 선분을 순차적으로 확인한다:

for each edge in polygon:

- 사용자의 위치가 현재 선분의 왼쪽 또는 오른쪽에 있는지 확인한다.
- 교차점이 있을 경우 crossCount를 1 증가시킨다.

3. 최종 교차 횟수를 확인한다:

if crossCount % 2 == 0:

 사용자는 구역 밖에 있음 (False)

else:

 사용자는 구역 안에 있음 (True)

4. 결과를 반환한다.

종료

③ 최단 경로 탐색 알고리즘

안전한 경로를 제공하기 위해 A* 알고리즘을 사용하여 최단 경로를 탐색한다. A* 알고리즘은 출발지에서 목적지까지의 경로를 탐색하면서, 각 노드에서 비용을 계산해 효율적이고 빠른 경로를 제공한다. 이 알고리즘은 휴리스틱 함수와 경로 비용을 결합하여 최적의 경로를 찾는다.

입력:

시작 노드 (start), 목표 노드 (goal), 장애물 좌표 (obstacles)

출력:

최단 경로 (path)

1. 시작 노드를 초기화한다:

openSet = {start}

closedSet = {}

2. 각 노드에 대한 $g(x)$ 와 $f(x)$ 를 초기화한다:

$g(\text{start}) = 0$ // 시작 노드의 $g(x)$ 는 0

$f(\text{start}) = g(\text{start}) + h(\text{start}, \text{goal})$ // h 는 휴리스틱 함수

3. 열린 리스트에서 $f(x)$ 가 가장 낮은 노드를 선택한다:

current = node in openSet with lowest $f(x)$

4. 목표 노드에 도착하면 경로를 반환한다:

if current == goal:

return 경로

5. 현재 노드를 열린 리스트에서 제거하고 닫힌 리스트에 추가한다:

openSet.remove(current)

closedSet.add(current)

6. 현재 노드의 인접 노드에 대해 다음을 수행한다:

for each neighbor of current:

if neighbor in closedSet:

continue // 이미 방문한 노드는 무시

```

tentative_g = g(current) + 거리(current, neighbor)

if neighbor not in openSet:
    openSet.add(neighbor) // 새로운 노드는 열린 리스트에
추가
    if tentative_g >= g(neighbor):
        continue // 더 나은 경로가 아니면 무시

    g(neighbor) = tentative_g
    f(neighbor) = g(neighbor) + h(neighbor, goal)
    경로를 업데이트한다

7. 반복하여 목표 노드에 도달할 때까지 경로를 탐색한다.

8. 경로를 반환한다.

종료

```

3.2.3. 운전 습관 피드백 기능

운전자의 운전 습관을 분석하고 피드백을 제공하는 기능이다. 운전 기록 데이터를 기반으로 운전 점수와 피드백을 제공한다.

(1) 운전 점수 측정

급정거, 안전거리 유지 위반, 신호 위반 횟수를 종합하여 통계 기반 점수 계산 알고리즘을 사용해 운전 점수를 산정한다. 산정된 점수는 대시보드에 제공되어 운전자는 자신의 운전 스타일을 확인할 수 있다.

입력: 급정거 횟수 (H), 안전거리 유지 위반 횟수 (D), 신호 위반 횟수 (S)
출력: 총 운전 점수 (T)

1. 각 항목별 점수 계산:

$S_{\text{safety}} = \text{calculateSafetyScore}(H, D)$

$S_{\text{law}} = \text{calculateLawAbidingScore}(S)$

$S_{\text{habit}} = \text{calculateHabitScore}(H)$

$S_{env} = \text{calculateEnvironmentResponseScore}(D, S)$

2. 총 점수 계산:

$$T = ((S_{safety} + S_{law} + S_{habit} + S_{env}) / (9 * 4)) * 100$$

3. 최종 점수를 0~100 범위로 제한:

$$T = \max(0, \min(T, 100))$$

4. 최종 점수 T를 반환

종료

(2) 운전 점수 그래프

급정거, 안전거리 유지 위반, 신호 위반 횟수를 기반으로 각 항목별 운전 점수를 계산한다. 시각화를 적용하여 운전자의 점수를 그래프로 제공함으로써, 각 항목에서의 성과를 시각적으로 쉽게 확인할 수 있도록 한다.

① 운전 안전성 점수 계산

입력:

H (급정거 횟수), D (안전거리 유지 위반 횟수)

출력:

S_{safety} (운전 안전성 점수)

1. 안전성 점수를 계산한다.

$$S_{safety} = 9 - (H * 1.5 + D * 1.0)$$

2. 점수를 1에서 9 사이로 제한한다.

$$S_{safety} = \max(1, \min(S_{safety}, 9))$$

3. 안전성 점수 S_{safety} 를 반환한다.

종료

② 교통 법규 준수 점수 계산

입력:

S (신호 위반 횟수)

출력:

S_law (교통 법규 준수 점수)

1. 법규 준수 점수를 계산한다.

$$S_law = 9 - (S * 2.0)$$

2. 점수를 1에서 9 사이로 제한한다.

$$S_law = \max(1, \min(S_law, 9))$$

3. 법규 준수 점수 S_law를 반환한다.

종료

③ 운전 습관 점수 계산

입력:

H (급정거 횟수)

출력:

S_habit (운전 습관 점수)

1. 운전 습관 점수를 계산한다.

$$S_habit = 9 - (H * 1.2)$$

2. 점수를 1에서 9 사이로 제한한다.

$$S_habit = \max(1, \min(S_habit, 9))$$

3. 운전 습관 점수 S_habit을 반환한다.

종료

④ 운전 환경 대응 점수 계산

입력:

D (안전거리 유지 위반 횟수), S (신호 위반 횟수)

출력:

S_env (운전 환경 대응 점수)

1. 환경 대응 점수를 계산한다.

$$S_{env} = 9 - ((D + S) * 0.8)$$

2. 점수를 1에서 9 사이로 제한한다.

$$S_{env} = \max(1, \min(S_{env}, 9))$$

3. 환경 대응 점수 S_env를 반환한다.

종료

(3) 운전 피드백

운전 중 발생한 급정거, 안전거리 유지 위반, 신호 위반 횟수를 바탕으로 운전 습관을 개선할 수 있는 맞춤형 피드백을 제공한다. 이를 통해 운전자는 더 안전하고 규칙적인 운전 습관을 형성할 수 있다.

(4) 최고 주행 거리 기록 제공

사용자의 주행 기록 중 최고 주행 거리를 계산하기 위해 거리 분석 알고리즘을 적용한다. 이 알고리즘은 두 지점 간의 위도와 경도를 기반으로 하여 하버사인 (Haversine) 공식을 사용해 거리(킬로미터)를 계산한다. 계산된 주행 거리 중 가장 긴 거리를 대시보드에 제공함으로써 운전자는 자신의 최고 기록을 확인하고 목표 주행 거리를 설정할 수 있다.

① 거리 분석 알고리즘 (Haversine Formula)

입력:

startLatitude (시작 지점의 위도), startLongitude (시작 지점의 경도),

endLatitude (종료 지점의 위도), endLongitude (종료 지점의 경도)
출력:

거리 (킬로미터 단위)

1. 위도와 경도의 차이를 계산한다:

$\text{deltaLatitude} = \text{Math.toRadians}(\text{endLatitude} - \text{startLatitude})$

$\text{deltaLongitude} = \text{Math.toRadians}(\text{endLongitude} - \text{startLongitude})$

2. 하버사인 공식을 적용하여 a 값을 계산한다:

$a = \sin(\text{deltaLatitude} / 2)^2 +$

$\sin(\text{deltaLongitude} / 2)^2 *$

$\cos(\text{Math.toRadians}(\text{startLatitude})) *$

$\cos(\text{Math.toRadians}(\text{endLatitude}))$

3. 대원의 중심각을 계산한다:

$c = 2 * \text{asin}(\text{sqrt}(a))$

4. 지구 반경(6371km)을 곱하여 최종 거리를 계산한다:

거리 = $\text{EARTH_RADIUS_KM} * c$

5. 계산된 거리를 반환한다.

종료

3.2.4. 긴급 상황 대응 기능

(1) 급발진 인식 기능

급발진 상황을 감지하면 즉시 112에 통화 연결을 시도하여 신속하게 신고할 수 있도록 도와준다. 이를 통해 운전자는 긴급 상황에 빠르게 대응할 수 있다.

(2) 교통사고 대처 방법 안내

교통사고 발생 시 필요한 대처 방법을 제공하여 운전자가 상황을 안전하게 처리할 수 있도록 안내한다. 적절한 대응을 통해 사고로 인한 추가적인 위험을 최소화할 수 있게 해준다.

4. 연구 결과 분석 및 평가

4.1. AI 모델 성능 평가

4.1.1. 차량, 사람, 신호등 탐지 (K_Traffic.pt)

YOLO v8 모델을 사용하여 도로 주행 환경에서 객체 탐지 성능을 개선하고자 총 9만 개의 이미지 데이터를 활용하여 차량, 사람, 신호등 등의 객체를 탐지하는 모델을 학습시켰다. 학습 결과는 아래와 같이 다양한 지표에서 시각적으로 표현되었다.

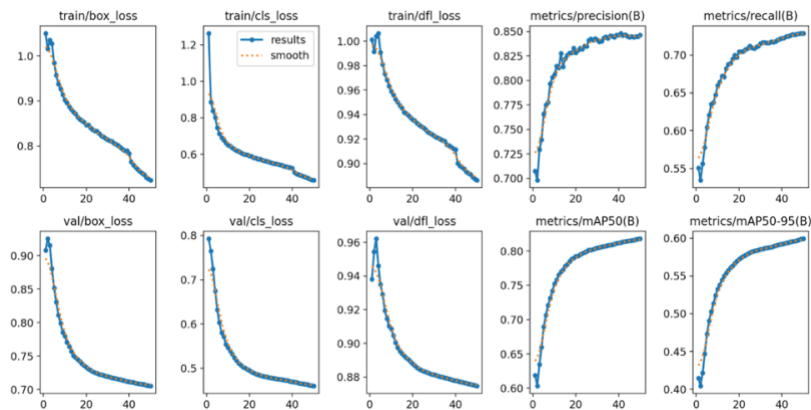


그림 5. YOLOv8 학습 결과

(1) 훈련 및 검증 손실

- train/box_loss: 바운딩 박스 예측의 손실은 학습 초반에 급격히 감소하며, 이후 점차적으로 완만해지면서 수렴하는 양상을 보였다. 이는 모델이 바운딩 박스의 위치를 점점 더 정확하게 예측하게 되었다는 것을 의미한다.
- train/cls_loss: 분류 손실 역시 학습이 진행됨에 따라 지속적으로 감소하였고, 이는 모델이 객체의 클래스(차량, 사람, 신호등 등)를 점점 더 정확하게 분류하고 있음을 나타낸다.
- train/dfl_loss: DFL(Distribution Focal Loss) 손실도 학습 초반 급격히 감소하고 수렴하는 경향을 보였다. 이는 객체 경계선에 대한 예측이 개선되고 있음을 시사한다.
- val/box_loss, val/cls_loss, val/dfl_loss: 검증 손실 또한 훈련 손실과 유사한 경향을 보였으며, 이는 모델이 훈련 데이터뿐만 아니라 검증 데이터에서도 잘 일반화되고 있음을 나타낸다.

(2) 평가 지표

- precision(B): 모델의 정밀도는 학습 초반부터 점차적으로 상승하며 0.85 이상에서 수렴하는 경향을 보였다. 이는 모델이 탐지한 객체 중에서 실제로 해당 클래스에 속하는 비율이 높아졌다는 것을 의미한다.
- recall(B): 재현율은 0.70에서 시작하여 점진적으로 향상되었으며, 약 0.75에서 수렴하였다. 이는 실제 객체 중에서 모델이 올바르게 탐지한 비율이 높아졌다는 것을 보여준다.
- mAP50(B): mAP50 값은 학습이 진행됨에 따라 꾸준히 증가하여 최종적으로 약 0.85에 도달하였다. 이는 모델이 50% 이상의 IoU(Intersection over Union)를 달성한 객체에 대해 높은 성능을 보였음을 시사한다.
- mAP50-95(B): mAP50-95 값은 다양한 IoU 기준에서의 모델 성능을 나타내며, 학습이 진행됨에 따라 점진적으로 개선되었다. 최종적으로 0.5에서 시작해 약 0.6에서 수렴하였다.

위의 결과들을 종합적으로 살펴보면, YOLO v8 모델이 차량, 사람, 신호등 등 다양한 객체를 효과적으로 탐지할 수 있음을 확인할 수 있었다. 또한 훈련과 검증 손실이 모두 안정적으로 감소하고, 정밀도, 재현율, mAP 등 주요 성능 지표들이 꾸준히 향상된 점에서, 본 모델이 도로 주행 환경에서의 객체 탐지에 충분한 성능을 발휘할 수 있음을 알 수 있다.

자율 주행을 위한 객체 탐지 성능을 실제 환경에서 비교하기 위해 실제 주행환경에서 두 가지 학습 버전(기본 YOLOv8 모델과 커스텀 데이터셋으로 학습된 YOLOv8)을 평가하였다.

기본 YOLOv8 모델은 일반적인 사전 학습된 데이터를 기반으로 차량을 탐지하였다. 해당 모델은 객체 탐지 성능이 상대적으로 우수하였으나, 특정한 환경(한국의 도로 상황, 차량 유형 등)에서의 정확도가 다소 낮게 나타났다. 예를 들어, 탐지된 차량의 신뢰도 점수는 0.6에서 0.8 정도로 측정되었다. 이로 인해 일부 작은 객체나 특수한 형태의 차량에 대해서는 인식률이 떨어질 수 있는 가능성이 확인되었다.



커스텀 데이터셋으로 학습된 YOLOv8 모델 K_Traffic은 한국의 도로 환경 및 다양한 차량 유형에 맞춘 데이터를 학습한 결과, 기본 모델보다 더 높은 정확도를 보여주었다. 신

회도 점수는 0.75에서 0.94로 나타나, 탐지된 차량에 대한 신뢰도가 더 높아졌다. 특히, 복잡한 교통 상황이나 우천등의 환경에서도 높은 탐지율을 보였으며, 차량의 위치와 형태를 보다 정확하게 탐지하는 성능을 발휘했다.



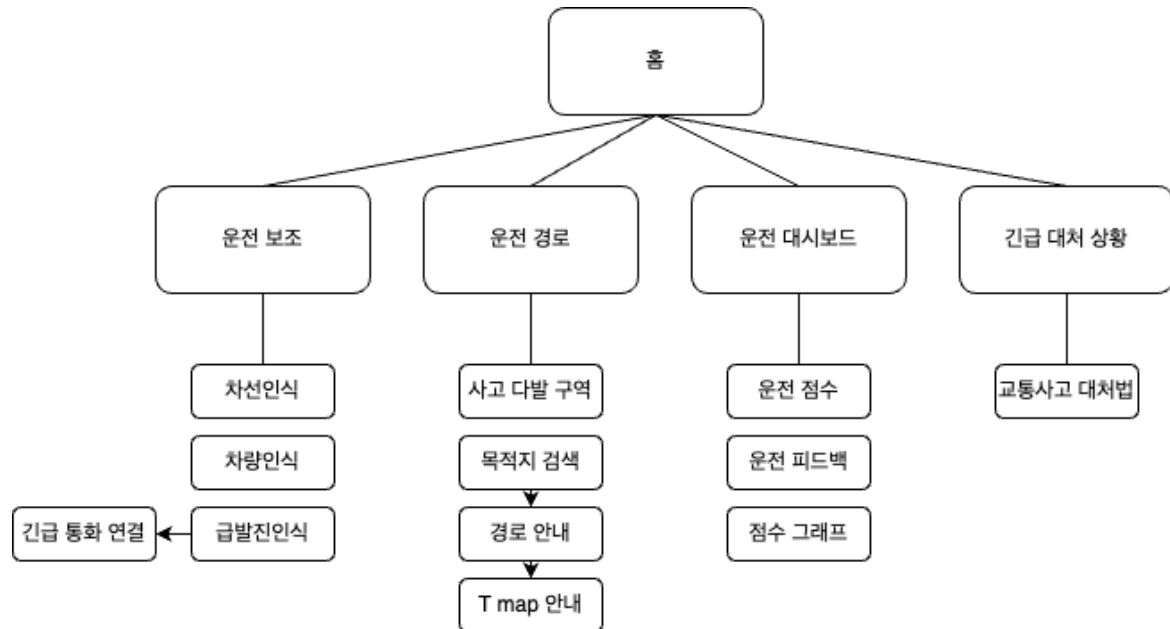
기본 모델로 신호등을 추론한 결과 이미지	K_Traffic으로 신호등을 추론한 결과 이미지
 A street view image showing a traffic light intersection. A white text box in the bottom left corner displays 'car 0.27'.	 A street view image showing a traffic light intersection. A pink text box in the center displays '초록초록불 0.63'.

4.1.2. 차선 탐지

차선 탐지 전	차선 탐지 후
 A first-person view from a car's dashboard looking out at a road intersection. A small inset in the top right shows a zoomed-in view of the road ahead. The road is clear.	 The same first-person view from the dashboard, but now a green cone-shaped overlay is projected onto the road ahead, indicating the detected lane.

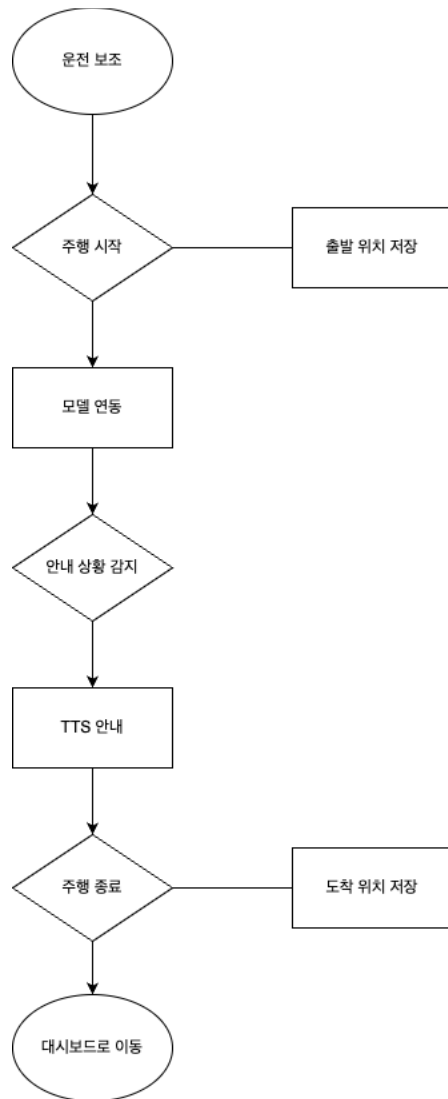
4.2. 서비스 구현

4.2.1. 기능 트리

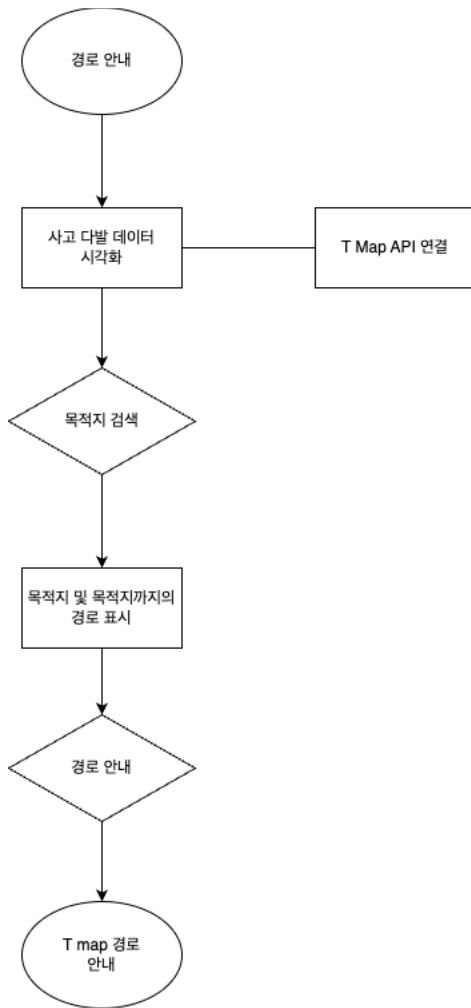


4.2.2. 플로우 차트

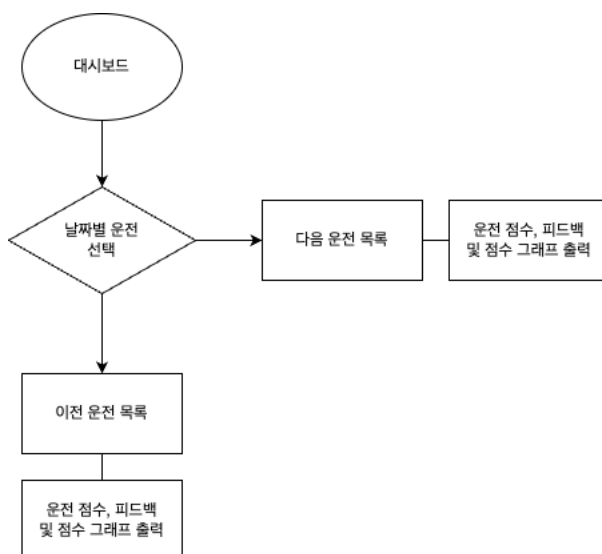
(1) 운전 보조 - CameraFragment



(2) 운전 경로 - MapFragment



(3) 운전 대시보드 - DashboardFragment



4.2.3. 유스 케이스

(1) 실시간 도로 상황 분석 및 경고 기능 제공

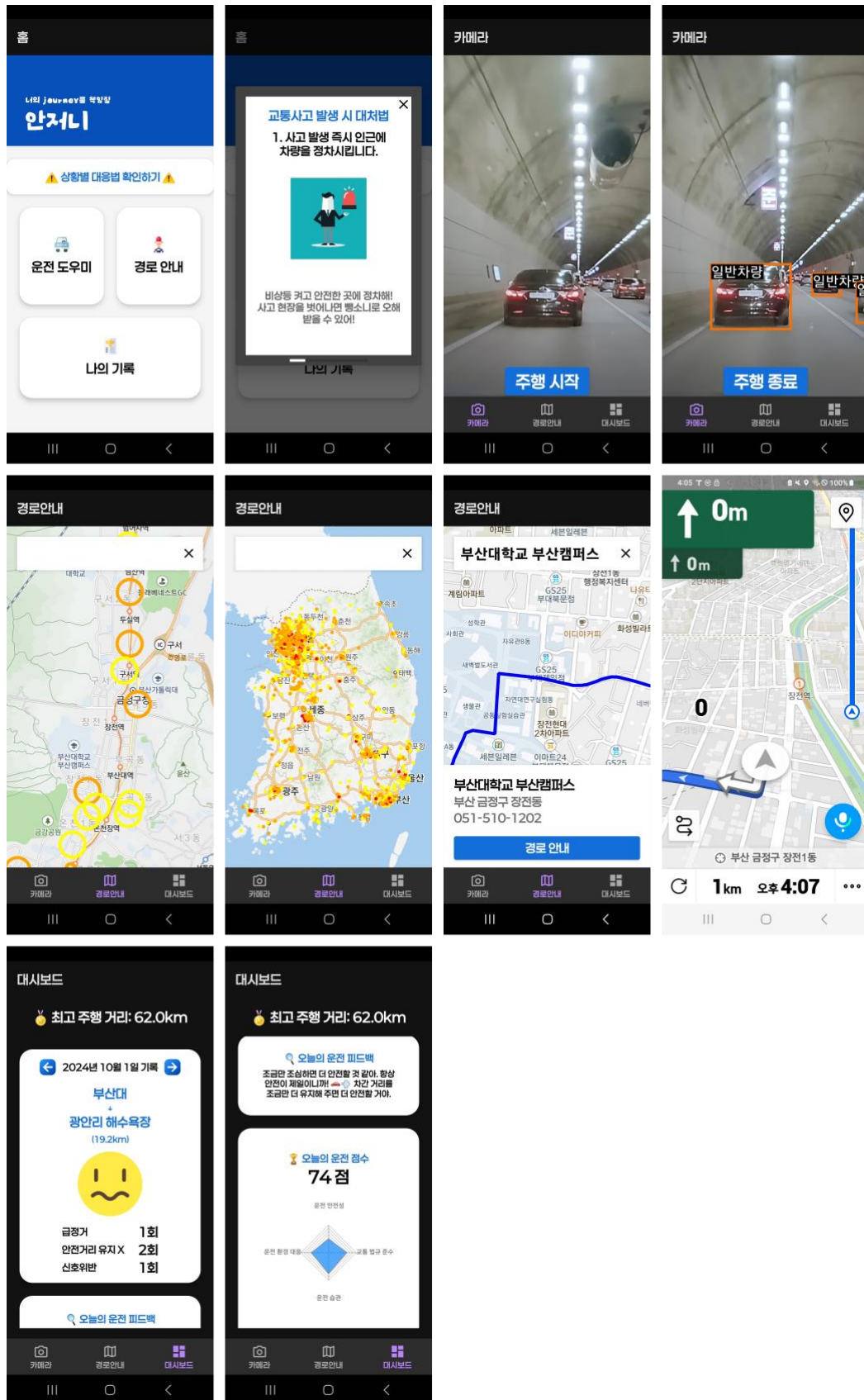
항목	설명
유스케이스 이름	실시간 도로 상황 분석 및 경고 기능 제공
목적	스마트폰의 후면 카메라를 사용하여 도로 상황을 실시간으로 분석하고, 위험 상황에서 운전자에게 경고를 제공한다.
액터	사용자
사전 조건	스마트폰의 후면 카메라가 정상적으로 작동하며, 앱이 카메라 권한을 가지고 있어야 한다.
후 조건	사용자는 실시간으로 도로 상황을 모니터링하며, 위험 상황에서 즉시 경고를 받는다.
주 시나리오	1. 사용자가 실시간 주행 모드로 전환한다. 2. 후면 카메라가 도로 상황을 분석한다. 3. 시스템이 차선 이탈, 안전거리 미확보, 신호 위반 등의 위험 상황을 감지한다. 4. 위험 상황이 발생하면 시스템이 음성 또는 시각적 경고를 제공한다.
확장 시나리오	1a. 카메라에 장애물이 가려진 경우: 시스템이 사용자에게 경고 메시지를 제공하고, 카메라 렌즈를 확인하도록 요청한다.
빈도	사용자는 주행할 때마다 실시간 도로 분석을 활성화할 수 있다.
특별 요구사항	카메라 영상 처리 속도는 실시간 경고를 제공할 수 있을 만큼 빠르고 정확해야 한다.

(2) 지도에서 경로 탐색 및 사고 다발 구역 표시

항목	설명
유스케이스 이름	지도에서 경로 탐색 및 사고 다발 구역 표시
목적	사용자가 사고 다발 구역을 피해 목적지까지 안전한 경로를 탐색하고 실시간 교통 정보 및 사고 구역을 확인한다.
액터	사용자
사전 조건	사용자가 목적지를 입력해야 하며, 시스템은 교통 사고 다발 구역 데이터를 확보해야 한다.
후 조건	사용자는 사고 다발 구역을 피한 안전한 경로를 확인하고,

	해당 경로를 따라갈 수 있다.
주 시나리오	1. 사용자가 목적지를 입력한다. 2. 시스템이 교통 사고 다발 구역 데이터를 기반으로 경로를 재탐색한다. 3. 시스템이 사고 다발 구역을 지도에 표시한다. 4. 사용자는 최적의 경로를 확인하고 경로 안내를 시작한다.
확장 시나리오	1a. 인터넷 연결이 끊긴 경우: 시스템이 사용자에게 네트워크 오류 메시지를 표시하고, 오프라인 상태로 경로를 탐색하지 못함을 알린다.
빈도	사용자는 주행이 완료될 때마다 운전 점수를 확인할 수 있다.
특별 요구사항	분석 결과는 정확하고 신뢰할 수 있어야 하며, 점수 산정 기준은 사용자에게 명확하게 전달되어야 한다.

4.2.4. 와이어 프레임



5. 결론 및 향후 연구 방향

본 연구에서는 운전 미숙자를 위한 운전 보조 애플리케이션을 개발하여, 차량 및 신호등 인식과 차선 인식 등의 기능을 통해 실시간 도로 상황 분석 기능을 제공했다. 이를 통해 운전자는 다양한 도로 상황에서 보다 안전하게 운전할 수 있도록 도움을 받을 수 있다. 연구 결과, AI 모델인 YOLOv8을 활용한 차량 및 신호등 인식 모델은 주간과 야간, 다양한 기후 조건에서도 일정 수준 이상의 성능을 보였으며, OpenCV를 이용한 차선 인식 모델은 도로의 차선을 안정적으로 인식하고 시각적으로 표시하는 데 효과적이었다.

차량 인식과 신호등 인식 모델은 AI Hub 데이터를 기반으로 학습되었으며, 커스텀 데이터셋을 활용하여 한국의 도로 환경에 최적화된 성능을 보여주었다. 특히, 다양한 객체를 정확하게 탐지하고, 실시간으로 변화하는 도로 상황에 빠르게 대응할 수 있었다. 그러나 야간이나 비가 오는 날씨 등 조도가 낮은 상황에서는 인식률이 다소 떨어지는 문제를 보였다. 이를 해결하기 위해 이미지 전처리 기법의 추가적인 적용과 데이터셋 확대가 필요하다.

차선 인식 모델은 OpenCV를 통해 구현하였으며, Canny Edge 검출과 Hough Line Transform을 이용해 차선을 감지하고 시각적으로 표시했다. 이 모델은 주간과 일반적인 도로 환경에서 높은 정확도를 나타냈지만, 급격한 곡선이나 도로 표면이 불규칙한 경우에는 성능이 저하되는 경향이 있었다. 이와 같은 문제를 개선하기 위해 더 복잡한 곡선 검출 알고리즘이나 머신러닝 기반의 차선 검출 기법을 추가적으로 도입할 수 있다.

향후 연구에서는 다음과 같은 방향으로 애플리케이션을 개선하고자 한다.

- 야간 및 악천후 환경에서의 인식 성능 향상: 현재 모델은 낮 시간대나 좋은 날씨 조건에서 우수한 성능을 보였으나, 야간이나 비, 눈 등 조도가 낮은 상황에서는 인식률이 떨어졌다. 이를 개선하기 위해 추가적인 데이터 수집과 함께, 이미지 보정 및 밝기 조절 등의 전처리 기법을 활용할 계획이다.
- 모델 경량화 및 최적화: YOLOv8 모델을 사용한 객체 탐지의 경우, 높은 정확도와 실시간 추론 속도를 유지하면서도 스마트폰과 같은 모바일 환경에서 효율적으로 작동할 수 있도록 모델의 경량화가 필요하다. 이를 위해 양자화(Quantization)나 모델 압축(Pruning) 기법을 적용하여 계산 속도를 향상시키고, 메모리 사용량을 줄일 계획이다.
- 다양한 도로 환경에 대한 데이터 증강: 한국 도로뿐만 아니라 다양한 지역과 조

건을 반영할 수 있는 데이터셋을 수집하고, 이를 활용하여 모델을 재학습시켜 범용성을 높일 예정이다. 특히, 도심지와 고속도로, 농촌 지역 등의 다양한 도로 환경을 반영한 데이터셋이 필요하다.

- 운전 피드백 기능의 고도화: 운전자의 습관을 보다 정밀하게 분석하고, 실시간 피드백을 제공할 수 있는 알고리즘을 개발할 예정이다. 예를 들어, 운전 중 과속 경향이나 급정거 빈도를 실시간으로 모니터링하여 피드백을 제공함으로써 운전자가 안전한 주행 습관을 형성할 수 있도록 돕는 방향으로 개선할 수 있다.
- 지능형 경로 안내 기능 개발: 기존의 사고 다발 구역을 회피하는 경로 탐색 기능을 한층 발전시켜, 실시간 교통 상황과 운전자의 운전 습관을 반영한 맞춤형 경로를 제안하는 시스템을 개발할 계획이다. 이를 통해 사용자에게 더욱 안전하고 효율적인 주행 경로를 안내할 수 있을 것이다.

6. 구성원별 역할 및 개발 일정

6.1. 구성원별 역할

6.1.1. 김대길

- 주변 차량, 보행자, 신호 인식을 위한 논문 탐색 및 관련 코드 분석
- 주변 차량, 보행자, 신호 인식을 위한 데이터셋 전처리 및 가공
- YOLOv8 모델의 성능과 구조에 대한 분석 및 연구
- YOLOv8 기반의 주변 차량, 보행자, 신호 인식 모델 설계 및 구현
- 탐지 결과를 화면에 표시하는 후처리 코드 구현

6.1.2. 김주송

- 애플리케이션 개발
- 피그마를 활용한 애플리케이션 UI 설계 및 구현
- 초보 운전자 연수 기능 설계
- 피드백 대시보드 기능 설계
- 긴급상황 대응 기능 설계

6.1.3. 이지수

- 차선 인식을 위한 논문 탐색 및 관련 코드 분석
- 차선 인식을 위한 데이터셋 전처리 및 가공
- YOLOv8 모델의 성능과 구조에 대한 분석 및 연구
- YOLOv8 기반의 차선 인식 모델 설계 및 구현
- 탐지 결과를 화면에 표시하는 후처리 코드 구현
- 피그마를 활용한 애플리케이션 UI 설계 및 구현

6.2. 개발 일정

5 월		6 월				7 월				8 월				9 월				10 월			
3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
연구																					
	데이터 수집	AI 모델 개발																			
	애플리케이션 개발																				
																기능 점검 및 유지 보수					

7. 참고 문헌

- [1] AI Hub. (2023, September). "Vehicle and Pedestrian Recognition Image" [Data set]. Available:
<https://www.aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&dataSetSn=195>
- [2] AI Hub. (2021, November). "Traffic Light/Road Sign Recognition Image (Seoul Metropolitan Area)" [Data set]. Available:
<https://www.aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&dataSetSn=188>
- [3] Ultralytics. (2024). "Ultralytics YOLO Documentation". Available:
<https://docs.ultralytics.com/>
- [4] Ultralytics. (2024). "ultralytics/ultralytics". GitHub. Available:
<https://github.com/ultralytics/ultralytics>