理學碩士學位論文

가상 데이터를 활용한 자율 주행 모델 성능 향상

仁濟大學校 大學院

컴퓨터應用科學科 器械學習 專攻

河 太 成

指導敎授 李 亨 源

가상 데이터를 활용한 자율 주행 모델 성능 향상

仁濟大學校 大學院

컴퓨터應用科學科 器械學習 專攻

河 太 成

이 論文을 理學碩士論文으로 提出함

指導敎授 李 亨 源

2023年 12月

河太成의 理學碩士論文을 認定함.

委員長 김 경 이 印

委 員 이 형 원 印

委 員 김 희 철 印

仁濟大學校 大學院

2023年 12月

목차

그림 목차

표 목차

수식 목차

국문 초록

가상 데이터를 활용한 자율 주행 모델 성능 향상

하 태 성

(지도교수 : 이학박사 이 형 원)

인제대학교 대학원 컴퓨터응용과학과

모델을 개발할 때 사용되는 데이터의 양과 품질은 모델의 성능에 결정적인 영향을 미친다. 자율 주행 같은 분야에서는 트랙의 선을 파악해 조향 각도를 결정하거나, 신호등 상태와 도로의 정지선을 인식하여 멈추거나 주행하는 작업에 다양한 도로 상황과 주변 환경은 필수적이다. 또한, 실제 주행 중에는 접하기 어려운 예기치 않은 상황에 대한 대처 데이터도 필요하다. 하지만 주행 데이터는 일반적으로 사용자가 직접 운전하며 얻게 되고, 돌발 상황을 인위적으로 구현하는 것에 한계가 있다. 더불어 특정 공간에서만 활용되는 모델의 경우, 한정된 데이터 수집이 이루어질 가능성이 높아지는데, 이러한 수집은 모델의 과적합을 초래하고, 새로운 환경에 적응하는 능력을 감소시킬 위험이 있다. 학습 데이터의 부족 문제를 보완하기 위해 본 논문에서는 가상 환경에서의 데이터 활용을 제안한다. 실험을 위해 실제 차량의 1/10 크기의 RC카와 모의 도로를 구축하였으며, 실제와 유사하면서도 다양한 환경을 제공하는 가상공간을 설계하였다.

실험은 크게 세 단계로 진행되었다. 첫 번째 실험에서는 합성 데이터와 실제 데이터의 성능을 확인하기 위해 데이터(차량의 정면 이미지, 차량의 조향 각도)와 모델(Vgg16, Mobile_Net_v2, Pilot_Net)을 준비하였고, 데이터와 모델별 성능 테스트를 진행하였다.

두 번째 실험에서는 실제 데이터와 가상 데이터 각각의 성능을 분석하기 위해 첫 번째 실험에서 우수한 성능을 나타낸 모델을 차량의 조향각도와 가속 여

부를 예측하도록 수정 후, 세 가지 다른 데이터 구성(실제 데이터만 100%, 가상 데이터만 100%, 실제 데이터100%와 가상 데이터 100%를 혼합시킨 데이터)을 사용하여 데이터별 성능 테스트를 진행하였다.

세 번째 실험에서는 두 번째 실험에서 학습된 모델(Mobile_Net_v2)의 실제 주행 성능을 평가하였다. 이를 위해 평가 지표를 개발하였고, 이를 바탕으로 모델이 실제 환경에서 얼마나 잘 주행하는지를 테스트하였다.

학습 데이터는 실제 데이터와 가상 데이터 각각 3만장을 수집하였고, 테스트 데이터는 실제 주행 데이터 1만5천장을 수집하였다. 실제 데이터 수집에는 약 4시간이 소요되었으며, 가상 데이터 수집에는 약 1시간이 걸렸다. 실제 데이터는 제한된 환경에서 수집되었지만, 가상공간에서는 1000장 단위로 환경을 바꿔가며, 하늘의 상태나 건물의 빛의 세기 등 다양한 환경에서 데이터를 획득할 수 있었다.

첫 번째 실험 결과 3가지 모델 중 Mobile_Net_v2와 Pilot_Net두 가지 모델에서 합성데이터를 사용했을 때 성능 향상을 보여주었고, 데이터 상관없이 성능만 비교했을 때 Vgg16모델이 제일 좋은 성능을 보여주었다. 하지만 Vgg16모델은 Jetson에서 20FPS로 예측이 힘들어 성능이 비슷하게 나온 Mobile_Net_v2모델을 사용하여 두 번째 실험을 진행하였다.

두 번째 실험 결과 Mobile_Net_v2에 세 가지 데이터 구성을 사용해 학습한 결과, 실제 데이터와 가상데이터를 합친 합성 데이터를 사용하였을 때 실제 데이터와 가상 데이터보다 좋은 성능을 보여주었다.

세 번째 실험결과 실제 데이터만 학습시킨 모델과 합성 데이터를 학습시킨 모델의 사고 비율은 달랐지만 점수 차이는 없었다.

Key word : 딥러닝, 강화학습, 자율 주행, 가상 데이터, 가상공간

ABSTRACT

Improving autonomous driving model performance using virtual data

Tae-seong Ha

(Advisor : prof, Hyung-Won Lee, Ph. D)

Department of Computer Aided Science

Graduate School, Inje University.

The quantity and quality of data used in developing a model critically influence the performance of the model. In areas such as autonomous driving, it is essential to discern the lines on the track to determine steering angles, and recognize the status of traffic lights and the stop lines on roads for stopping or continuing to drive. Various road conditions and surrounding environments are imperative in these processes. Additionally, dealing with unexpected situations that are hard to encounter during actual driving requires relevant data. However, driving data typically comes from users driving personally, and there are limits to artificially creating sudden incidents. Moreover, in the case of models used in specific spaces, there is a high likelihood of limited data collection occurring. Such collection might lead to overfitting of the model and pose risks of reducing the model's adaptability to new environments. To compensate for the lack of training data, this paper proposes the utilization of data from virtual environments. For experimentation, a road for RC cars, one-tenth the size of actual vehicles, was constructed, and a virtual space offering diverse environments while being similar to reality was designed.

The experiment was conducted in three major stages. In the first

experiment, synthetic data and real data's performance were evaluated. For this, data (front images of the vehicle, steering angles of the vehicle) and models (Vgg16, Mobile_Net_v2, Pilot_Net) were prepared, and performance tests were conducted according to each data set and model.

In the second experiment, the performance of real and virtual data was analyzed individually. The model exhibiting superior performance in the first experiment was modified to predict the vehicle's steering angle and acceleration status. Performance tests were conducted using three different data configurations: solely real data (100%), solely virtual data (100%), and a mixture of real (100%) and virtual data (100%).

In the third experiment, the actual driving performance of the learned model (Mobile_Net_v2) from the second experiment was evaluated. Evaluation metrics were developed for this purpose, and based on these, tests were conducted to assess how well the model performs in real-world driving scenarios.

Training data consisted of 30,000 images each of real and virtual data, and 15,000 images of actual driving data were collected for testing. Collecting real data took about four hours, while virtual data collection took about one hour. Although real data was collected in a limited environment, various environmental data could be acquired in the virtual space by changing environments every 1000 images, such as the sky's condition and the intensity of building lights.

The results of the first experiment showed that performance improvement was seen in two of the three models, Mobile_Net_v2 and Pilot_Net, when synthetic data was used. Comparing performance alone, regardless of the data, the Vgg16 model showed the best performance. However, as predicting at 20 FPS was challenging on the Vgg16 model on Jetson, the second experiment proceeded using the Mobile_Net_v2 model, which showed similar performance.

In the results of the second experiment, after training the Mobile_Net_v2 with three different data configurations, the model using a combination of real and virtual data (synthetic data) showed better performance than using real or virtual data alone.

In the results of the third experiment, there was no difference in scores between the model trained only with real data and the model trained with synthetic data, but there was a difference in the accident rate.

Key word : Deep Learning, Reinforcement Learning, Autonomous Driving, Virtual Data, Virtual Space

Ⅰ. 서론

자율 주행 기술은 현재 CNN, RNN 등 다양한 모델을 활용하여 연구와 기술 발전이 활발하게 진행되고 있는 분야이다. 특히 CNN 기반 모델은 이미지에 내재된 풍부한 정보로 차량과 신호등, 보행자, 도로 표시 등 중요한 객체들을 정확하게 인식하고, 다양한 조명과 기상 조건, 그리고 다양한 시점에서도 객체를 안정적으로 인식하는 능력 때문에 광범위하게 사용되고 있다. 그러나 좋은 성능을 지닌 모델을 개발하기 위해서는 방대한 데이터와 최적화된 모델 구조가 필수적이다.[1] 실제 도로 환경에서의 데이터 수집은 비용과 시간적인 문제로 어려움이 있으며, 특히 위험한 상황이나 예외적인 경우에 대한 데이터는 더욱 힘들게 접근하기 마련이다. 이러한 한계로 인해, CNN 모델은 특정 환경에 대해 과적합될 수 있으며, 이는 다양한 상황에 적절히 대응하는데 제약을 가져올 수 있다.

CNN 모델의 성능을 향상시키기 위한 대안으로, 기존 데이터를 회전, 확대/축소, 반전 등의 변형을 통해 데이터의 양을 인위적으로 확장하는 기법인 데이터 증강(Data Augmentation)이 있다. 또한, 미리 대규모 데이터 셋에서 학습된 모델의 가중치를 사용하여 작은 데이터 셋에 대한 학습을 진행하는 전이학습(Transfer Learning)도 효과적이다.[2] 최근에는 Domain Randomization기법과 같이 가상 환경에서 생성된 데이터를 활용하는 연구도 활발히 진행되고 있다.

본 논문에서는 가상 환경의 연구 동향을 바탕으로, 실제로 1/10크기의 RC카와 제한적인 주행 환경, 가상 환경을 제작하여 가상데이터의 효과를 알아보고자 한다.

Ⅱ. 연구 목적

기술 발전에 따라, CNN (Convolutional Neural Network) 기반의 모델은 그 중요성이 더욱 부각되고 있다. 이러한 모델들은 다양한 환경과 상황에서의 적응성이 필수적으로 요구되며, 이를 위해 지도학습이 주로 활용되고 있다. 이 지도학습 방식은 품질 높은 입력 데이터와 그에 해당하는 정답 데이터를 기반으로 모델의 성능을 꾸준히 향상시키는 방향으로 연구가 진행되어 왔다. 하지만, 이러한 방법에도 불구하고 적절한 모델을 선별하기 위한 다양한 실험과 특히 대규모의 고품질 데이터 수집이 필수적인데, 이는 시간과 비용이 많이 소요되는 큰 장벽으로 작용하곤 한다.

본 연구는 위와 같은 문제점을 극복하기 위한 방안으로, 가상 데이터를 활용하는 접근법을 제시하고자 한다. 이를 통해 제한된 공간 및 조건에서도 자율 주행 모델의 학습 성능을 향상시킬 수 있을 것으로 기대된다. 더불어, 실시간 자율 주행에 적용 가능한 다양한 CNN 모델 후보들을 테스트하며, 자율 주행에 가장 적합한 모델을 도출하고자 한다.

만약 혼합 데이터, 즉 가상데이터와 실제 데이터의 조합이 우수한 성능을 나타낸다면, 이는 의료, 우주 분야 등에서 흔히 마주하는 데이터 부족 문제의 해결책으로 크게 기여할 것으로 기대된다.

Ⅲ. 가상 데이터에 관련된 선행 연구

1. Virtual Worlds as Proxy for Multi-Object Tracking Analysis

논문에 나오는 VKITTI 데이터 셋은 KITTI 데이터 셋을 기반으로 제작된 가상 환경의 데이터 셋이다. KITTI 데이터 셋은 실제 차량을 활용하여 도로를 주행하면서, 차량에 장착된 스테레오 카메라, LiDAR, GPS, IMU와 같은 센서들로부터 데이터를 수집하며, 수집한 데이터 셋은 광학 흐름, 스테레오 깊이, 3D 물체 탐지 등의 다양한 작업에 대한 벤치마크를 제공한다. (그림 1)

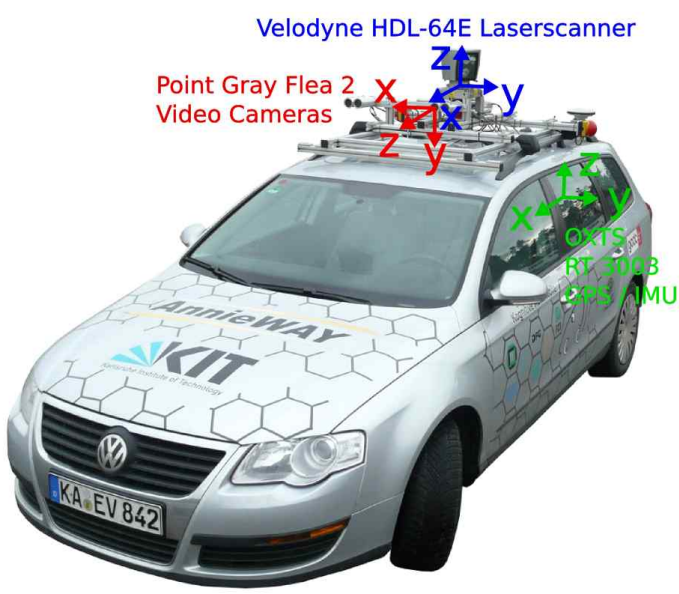

그림 1 : 차량 상단에 GPS, 레이저 스캐너, 카메라를 통해 데이터를 수집한다.

그러나 KITTI 데이터 셋에는 다양한 환경 조건(비, 안개, 눈)과 다양한 도로 상황, 교통 상황 등에 대한 데이터가 충분치 않다는 문제점이 있었고, 이를 해결하기 위해, Virtual Worlds as Proxy for Multi-Object Tracking Analysis 논문에서 그림 2에서 볼 수 있듯이 KITTI 데이터 셋을 바탕으로 가상 환경을 구축하며, 다양한 요소들을 조절하여 데이터의 양을 확장하고 모델을 학습하였다. 연구에서는 DP-MCF와 MDP 방식을 사용하여 실험을 진행하였다. 이때, 실제 데

이터만을 활용한 경우(r), 가상 데이터를 활용한 경우(v), 그리고 가상 데이터로 학습한 후 실제 데이터로 재학습을 진행한 경우(v->r)로 실험을 구분하여 진행하였다. 실험 결과, 표 1에 따르면 DP-MCF에서는 실제 데이터만을 활용했을 때보다 v->r로 학습을 했을 때 4.8%의 성능 향상이 관찰되었고, MDP의 경우에는 0.6%의 성능 향상이 나타났다. 이를 통해, 가상 데이터와 실제 데이터의 혼합 사용이 모델의 성능 향상에 긍정적인 영향을 미친다는 것을 확인할 수 있었다.[3]

그림 2 : 좌측은 실제 공간 데이터, 우측은 가상공간 데이터

	MOTA↑	MOTP↑	MT↑	ML↑	I↓	F↓	P↑	R↑
DP-MCF v	64.3%	75.3%	35.9%	31.5%	0	15	96.6%	71.0%
DP-MCF r	71.9%	79.2%	45.0%	24.4%	5	17	98.0%	76.5%
DP-MCF v→r	76.7%	80.9%	53.2%	12.3%	7	27	98.3%	81.1%
MDP v	63.7%	75.5%	35.9%	36.9%	5	12	96.0%	70.6%
MDP r	78.1%	79.2%	60.7%	22.0%	3	9	97.3%	82.5%
MDP v→r	78.7%	80.0%	51.7%	19.4%	5	10	98.3%	82.6%

표 1 : 모델과 데이터차이에 따른 성능 표

2. Training Deep Networks with Synthetic Data

가상 데이터를 활용하여 모델의 성능을 향상시키는 접근 방식에 대한 또 다른 예시로 ‘Training Deep Networks with Synthetic Data Bridging the Reality Gap by Domain Randomization’ 논문을 들 수 있다. 논문에서는 이미지 분류 작업부터 이미지 분할 작업까지 다양하고 상당한 자원이 필요하다는 문제점을 제기한다. 이 문제를 극복하기 위해 논문의 저자들은 객체 외부의 배경 환경을 무작위로 변형함으로써 모델이 이미지의 주요 특징에 더욱 집중하도록 유도하는 방법론을 제안한다.

데이터 처리 과정은 임의의 배경 이미지를 선택하고, 해당 배경 위에 원하는 객체의 다양한 버전을 배치한다. 그 후, 객체의 질감, 위치, 빛의 반사와 같은 요소들을 무작위로 조정하여 사실적이지 않은 가상의 데이터를 생성한다. 연구에서는 위와 같은 방법으로 생성된 DR 데이터 셋 10만 장과 VKITTI 데이터 셋 2500장을 Faster R-CNN, R-FCN, 그리고 SSD 모델 학습에 활용하였다.(그림 3)

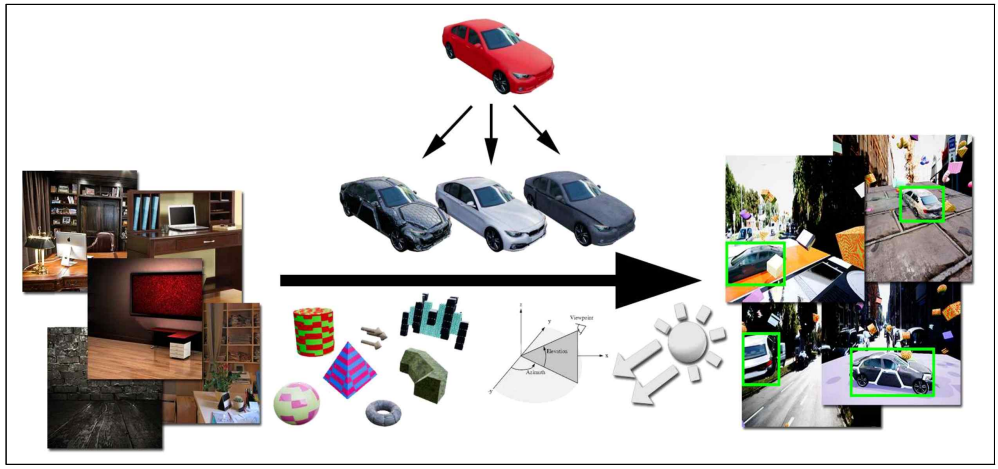

그림 3 : 배경, 질감, 빛의 방향등이 무작위로 선택되어 가상 데이터를 생성.

학습 결과, 표 2에서 보이는 것처럼 세 개의 모델 중에 R-FCN과 SSD 모델에서 뛰어난 성능을 보였다. 그림 4는 가상 데이터와 실제 데이터의 조합이 실제 데이터만 사용했을 때 보다 모델 성능 향상에 긍정적인 영향을 주는 것을 확인할 수 있었다. 또한 논문의 그림 3과 표 2를 통해 사실적이지 않은 데이터가 모델 성능 향상에 긍정적인 영향을 주는 것을 확인할 수 있었다.[4]

Architecture	*VKITTI* [7]	*DR* (ours)
Faster R-CNN [25]	**79.7**	78.1
R-FCN [2]	64.6	**71.5**
SSD [17]	36.1	**46.3**

표 2 : R-FCN과 SSD모델에서 DR를 사용했을 때 좋은 성능을 보인다.

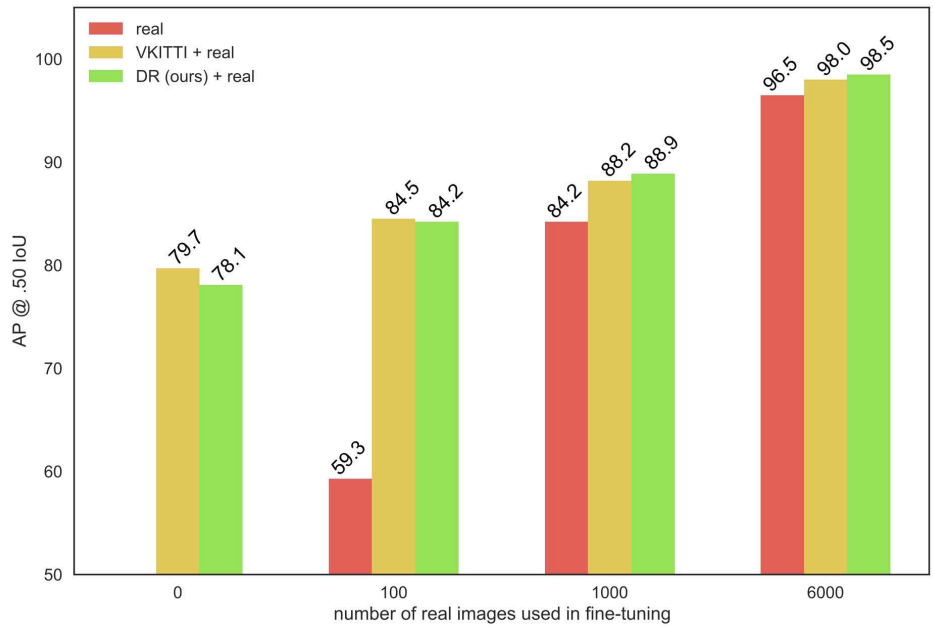

그림 4 : Faster R-CNN을 사용하여 가상 데이터 성능 비교

3. Driving in the Matrix Can Virtual Worlds Replace Human-Generated Annotations for Real World Tasks?

기존에 가상 데이터 수집 논문들은 현실 공간을 가상환경에 구현한 다음 환경을 조절하면서 데이터를 수집하거나, 가상공간에 새로운 객체를 생성하고, 환경을 수정하면서 데이터를 수집하였다. 이러한 수집 과정은 중관 관리 작업이 필요하다는 단점이 있다. “Driving in the Matrix: Can Virtual Worlds Replace Human-Generated Annotations for Real World Tasks?” 논문은 이러한 문제를 해결하기 위해, 현실적인 그래픽을 자랑하는 GTA5 게임을 활용하여 Sim 데이터 수집을 제안한다. 그림 5에서 보이는 방법은, GTA5 게임에 이미지 캡처용 플러그인을 설치하여 데이터를 수집하는 방식이다. 이 방법을 통해 낮, 밤, 아침, 황혼 등 다양한 시간대의 데이터를 시뮬레이션 할 수 있으며, 태양빛, 안개, 비 등의 요소도 미세 조정이 가능하다. 특히, GTA 게임 내부의 알고리즘을 활용하여 bounding box 데이터부터 segmentation 데이터까지 손쉽게 얻을 수 있었다고 한다.

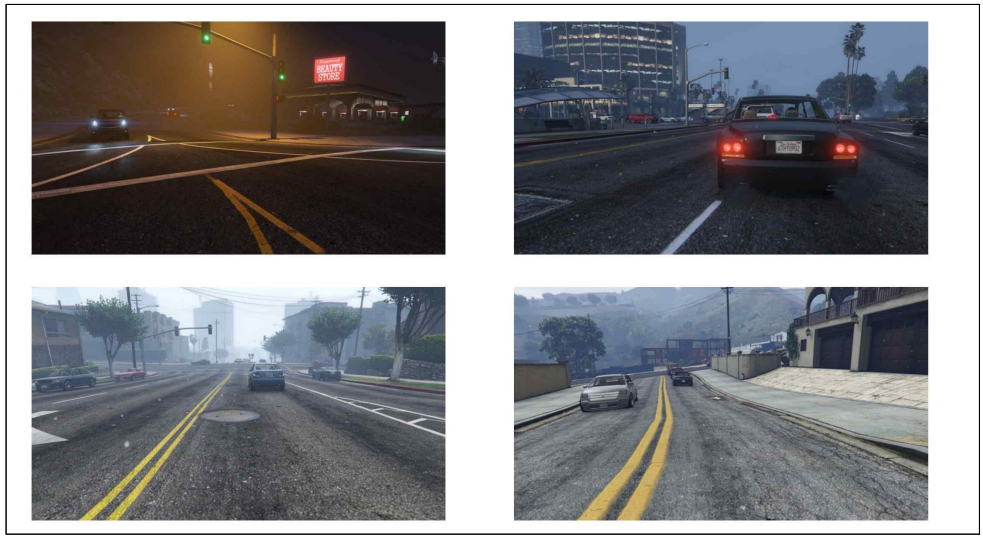

그림 5 : GTA5게임의 차량 주행 화면

그림 6에서 보이는 것처럼 논문에서 모델을 학습할 때 가상 데이터의 장점을

살려 현실 데이터 셋인 Cityscapes보다 훨씬 많은 데이터를 사용하였고, 성능 평가는 KITTI 데이터 셋을 사용하여 평가하였다. 모델의 성능은 Faster-R-CNN을 통해 평가되었으며, 그림 7의 결과에 따르면, Sim 데이터가 5만 장을 넘어가는 시점부터 현실 데이터보다 더 뛰어난 성능을 나타냈다.[5]

Data set	**# of images**
Cityscapes [20]	2,975
KITTI [6]	7,481
Sim 10k	10,000
Sim 50k	50,000
Sim 200k	200,000

그림 6 : Cityscapes, KITTI는 실제 데이터, Sim은 가상 데이터

Data set	**Easy**	**Moderate**	**Hard**
Sim 10k	0.5542	0.3828	0.2904
Sim 50k	**0.6856**	0.5008	0.3926
Sim 200k	0.6803	**0.5257**	**0.4207**
Cityscapes [20]	0.6247	0.4274	0.3566

그림 7 : 가상 데이터 성능표

Ⅳ. 자율 주행에 대한 선행 연구

1. End to End Learning for Self-Driving Cars

“End to End Learning for Self-Driving Cars“는 NVIDIA의 연구 팀이 발표한 논문으로, 자율 주행 자동차의 핵심 기술에 대한 새로운 접근 방식을 제시하였다. 전통적인 자율 주행 시스템은 센서 융합, 환경 인식, 경로 계획과 같은 다양한 과정으로 구성되어 있었으나, 이 논문에서는 카메라로부터 바로 얻은 이미지만을 입력으로 사용하여 차량의 조향각을 직접 예측하는 end-to-end 신경망 모델을 제안하였다.

이 모델의 아키텍처는 그림 8에서 확인할 수 있으며, 기본적으로는 컨볼루션 신경망(CNN) 구조를 따르고 있다. 특별히, 입력 이미지는 일반적인 CNN모델과는 다르게 66x200 크기의 YUV 컬러 스페이스로 변환되어 사용되며, 풀링층을 포함하지 않는 대신 바로 이미지를 정규화하는 과정을 거친다.

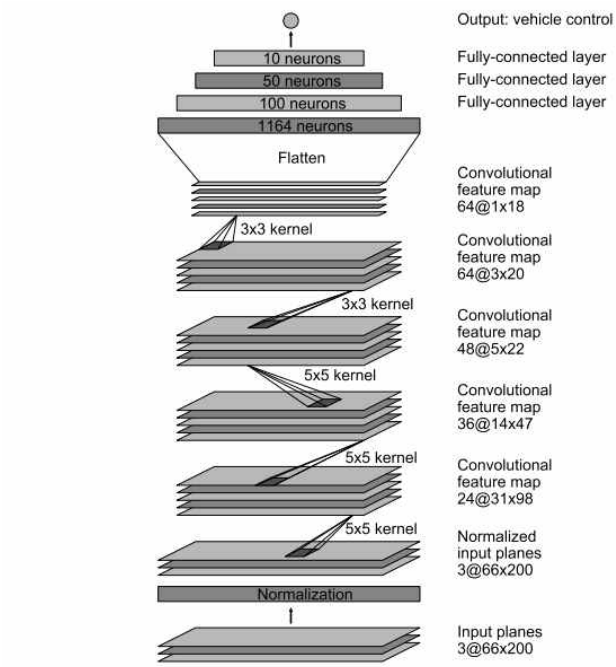

그림 8 : Pilot_net 구조

논문에서는 차량에 왼쪽, 오른쪽, 그리고 정면의 총 세 군데에 카메라를 설치하여 진행하였다. 이렇게 설치된 각 카메라는 정면 방향을 향하게 돌려져, 동시에 세 군데의 지점에서 데이터를 수집하였다. 데이터 수집에 있어서는, 직선 도로에서는 10FPS의 주기로 데이터를 수집했고, 곡선 도로에서는 프레임 수를 늘려서 데이터의 양을 더 많이 취득했다.

Pilot_net의 학습 과정은 그림 9처럼 세 군데에서 수집된 이미지 데이터를 간단한 전처리 과정을 거쳐서 차량의 조향 각도를 예측하도록 네트워크를 학습시켰고, 최종적으로 학습이 완료된 모델은 중앙 카메라에서만 얻은 이미지 정보를 활용하여 조향 각도를 성공적으로 예측하였다.

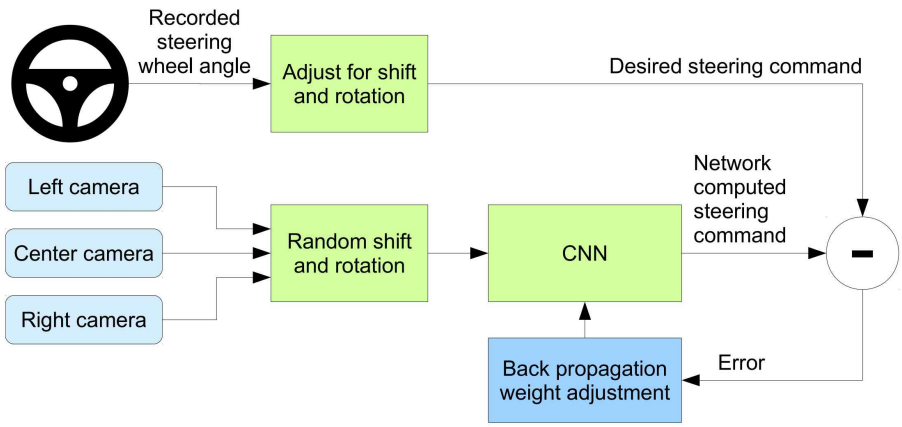

그림 9 : Pilot_net의 학습 과정

실제 도로에서의 테스트 전, 연구팀은 가상 환경에서 모델을 먼저 시험한 뒤 실제 환경에서 테스트를 진행하였다. 약 16킬로미터의 거리를 주행하는 도중, 전체 시간의 98% 동안 안정적으로 주행하는 능력을 보였고, 완성된 모델로 비포장도로와 숲 지역에서의 이미지를 촬영하여 특징 추출 및 상태 시각화를 진행하였다. 그 결과 그림 10에서 보이는 것처럼, 도로 위는 잘 감지되었으나, 숲과 같은 복잡한 환경에서는 의미 있는 특징 추출에 어려움을 겪었다는 것을 확인할 수 있었다.[6]

그림 10 : (a)은 도로, (b)은 숲을 보고 차선감지 진행

2. Autonomous Driving Simulation of Model Car Using PilotNet

“Autonomous Driving Simulation of Model Car(HENES-T870)Using PilotNet” 논문에서는 엔비디아의 Pilot_net을 이용하여 모형 차량의 자율 주행을 실험하였다. 논문에서는 실험용 모형 차량을 작동시키기 위해 그림 11와 같이 nucleo board를 사용하였으며, nucleo보드에서 이미지를 처리할 수 없어 PC로 차량 정면의 영상 정보를 전달하고 PC에서 연산처리 후 PC에서 받은 값을 차량에 전달하는 방식을 사용하였다.

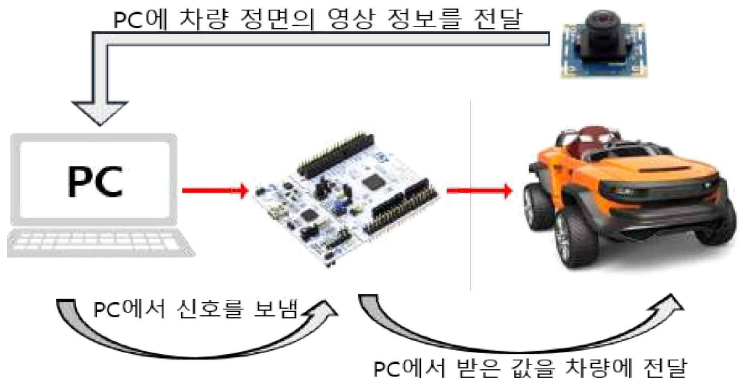

그림 11 : 논문에서 차량을 작동시키는 과정

Pilot_net 학습을 위해 400m 달리기 트랙에서 차량의 조향 각도만 조절하며 라인을 따라 주행하였다. 직선 구간의 데이터가 과다하게 수집되었기 때문에, 커브 구간에서 추가적으로 데이터를 수집하였다. 결과적으로 일반 주행 이미지는 4만장, 커브 주행 이미지는 1만 5천장으로, 총 5만 5천장의 데이터를 확보하였다.

학습에 앞서 이미지의 크기는 320x160으로 조절하였고, 차량의 조향 각도는 -20에서 +20의 범위를 -1에서 +1 사이의 값으로 정규화 하였다. 학습 시 사용한 손실 함수는 MSE였고, 총 5번의 테스트 결과 5개 모두 낮은 MSE값을 나타내었다.[7]

Ⅴ. 실험 환경 및 실험 데이터

1. 실제 공간

본 연구에서 사용된 차량의 디자인 및 구조는 그림 12에 제시되어 있다. 이 차량은 1/10 비율로 실제 도로에서의 주행 데이터 수집과 학습된 모델의 테스트 목적으로 설계되었다. 자율 주행 시스템에서 중심적인 연산을 담당하는 기기는 NVIDIA의 Jetson Xavier NX로, 이 장치의 활용으로 실제 차량의 동작을 보다 정확하게 모방할 수 있었다.

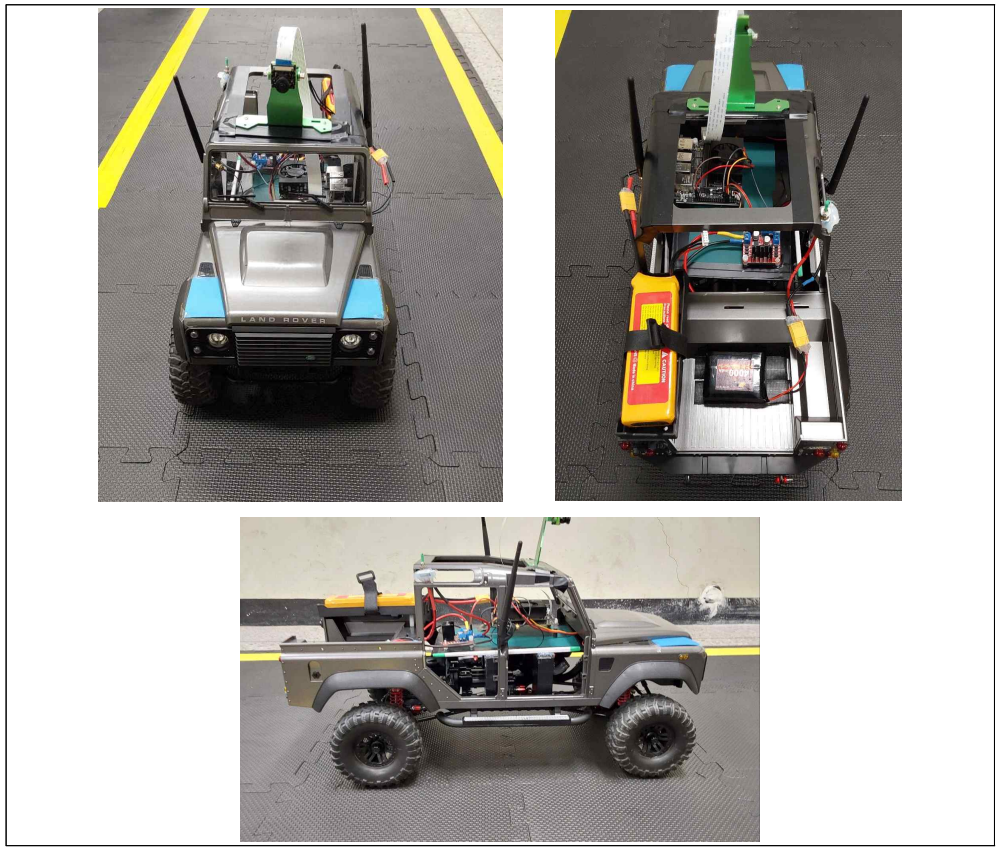

그림 12 : 카메라는 차량 상단면에 부착하였고, Jetson과 모터드라이버는 차량 내부, 배터리는 차량 뒷면에 부착하였다.

차량의 조향 제어는 Jetson의 'Jetson.GPIO' 라이브러리를 활용하여 이루어졌고, 모터의 회전 속도는 Jetson의 PWM 신호와 함께 모터 드라이버(N298N)를 사용하여 세밀하게 조절하였다. 본 연구에서는 차량의 정면 이미지를 수집하기 위한 카메라 선택 시 여러 기준을 고려하였다. 차량 정면 이미지에서 신호등이나 횡단보도와 같은 중요한 주행 정보가 포함될 수 있도록 넓은 시야각을 제공하는 것이 중요하다고 판단하였다. 또한, 1280x720의 해상도와 30FPS의 프레임 속도를 제공할 수 있어야 하며, 영상 데이터의 신속하고 안정적인 전송이 가능한 CSI 기술이 탑재된 카메라가 필요하다고 생각하였다. 이러한 조건을 만족하는 Sony IMX219 Sensor 카메라 모듈을 선택하였고, 선택된 카메라로 수집된 이미지는 Opencv를 활용하여 224x224 크기로 변환하여 데이터를 수집하였다.

추가적으로 데이터 수집의 효율성과 차량 조작의 편의를 위해 조이스틱 및 와이파이 모듈를 추가하였다. 이러한 추가 장치들은 차량을 더욱 쉽게 제어하고, 다양한 상황에서의 데이터를 효과적으로 수집할 수 있도록 도와주었다.

본 연구에서 완성한 RC카의 도로는 그림 13에 나타난 바와 같이 좌회전, 직진, 우회전의 세 가지 주행 방향을 고려하여 디자인되었다.

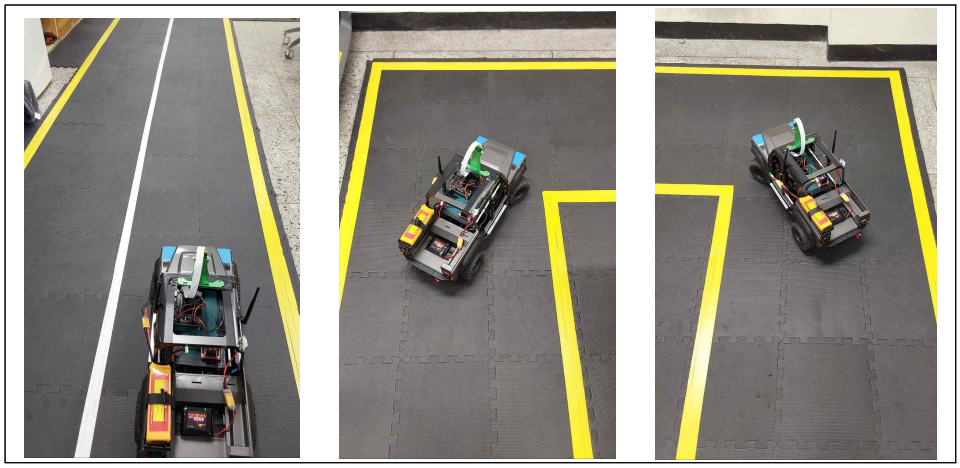

그림 13 : 직진 차선, 좌회전 차선, 우회전 차선

실제 도로의 환경을 묘사하기 위해 그림 14에서 보이는 3방향 교차로와 차선 변경 구관을 제작하였으며, 교차로 내부에는 차량의 움직임을 정밀하게 판단할 수 있도록 신호등과 유도선을 설치하였다.

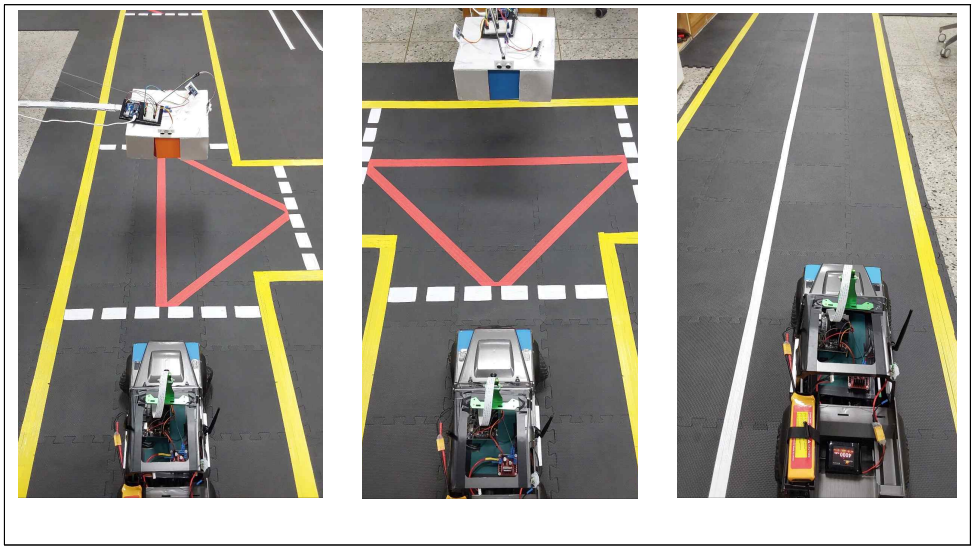

그림 14 : 유도선(빨간색), 신호등, 횡단보도

유도선은 차량의 주행 방향을 안내하며, 그림 15와 같이 카메라 화면에서 신호등이 일시적으로 보이지 않을 경우 차량의 움직임을 안내하기 위한 추가적인 참 조선으로 사용되었다.

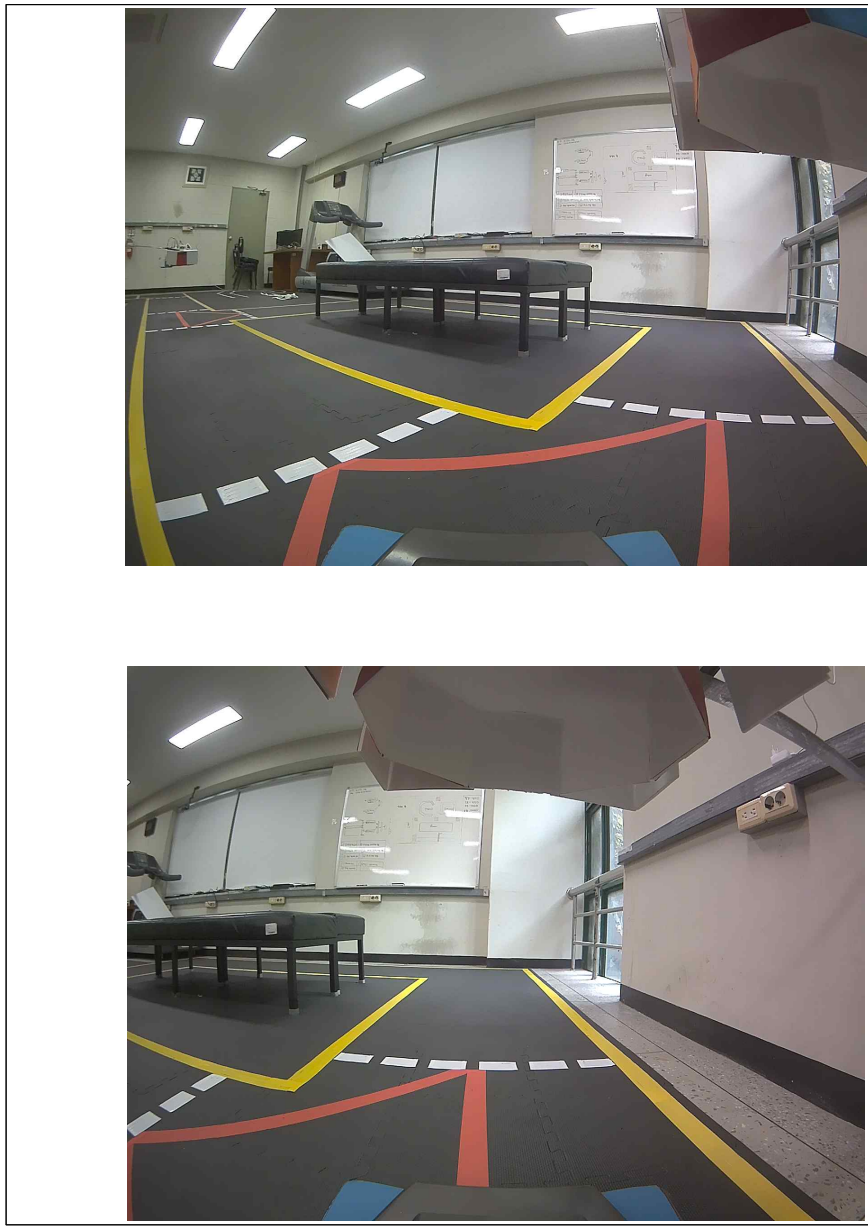

그림 15 : 좌측 이미지는 좌회전 주행 중, 우측 이미지는 직진 주행 중

신호등은 그림 14에 보이는 것처럼 제작하였으며, 네 가지 주요 신호를 표시할 수 있도록 디자인하였다. 정지(빨간색), 직진(초록색), 좌회전(파란색), 및 우회전(주황색)으로 구성되어 있으며, 신호등은 차량이 근접할 때까지 항상 정지 신호를 출력하다가 차량이 정지선에 도달하면 초음파 센서가 이를 감지한다. 감지된 이후, 신호등은 차량에게 랜덤한 주행 방향을 나타내는 신호를 보여주도록 제작되었다. 이러한 시스템은 실제 도로 상황에서의 차량의 반응을 더욱 정확하게 평가하고자 하는 목적으로 도입하였다.

2. 가상공간

가상 데이터 수집을 위한 공간을 제작하는 데에는 '유니티'를 활용하였다. 가상공간은 그림 16에서 볼 수 있듯이 신호등, 차선, 차량 크기, 카메라의 해상도 및 위치 등이 실제 환경과 유사하게 구성되었다. 특히 차량은 휠 콜라이더를 사용하여 기존에 유니티 기능 중 하나인 AddForce로 차량을 움직이는 것보다 실제 차량과 비슷하게 움직일 수 있도록 제작하였다.

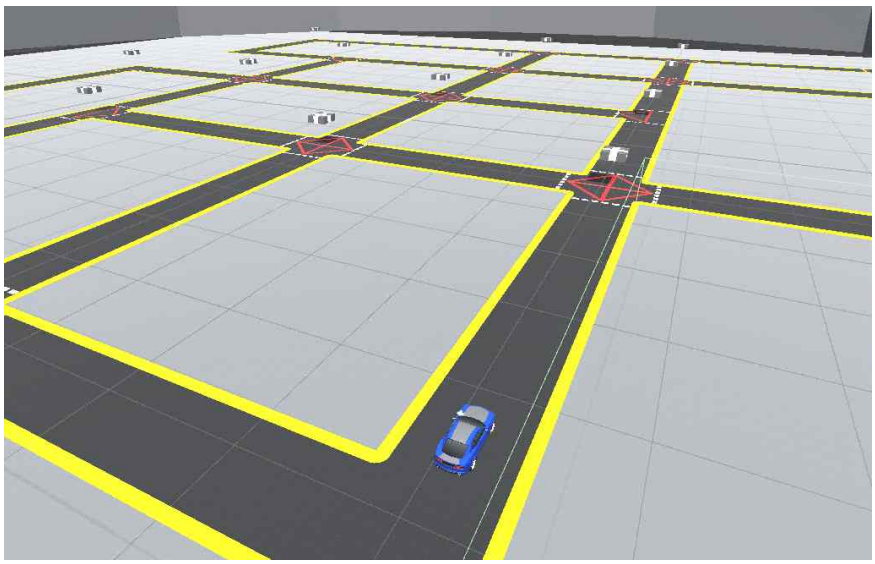

그림 16 : 실제 환경에 맞게 제작된 가상공간

데이터의 풍부한 다양성을 확보하기 위해서, 그림 17과 같이 환경 내의 요소들이 특정 시간 간격으로 자동 변경될 수 있도록 설계되었다. 이에는 하늘의 모습부터 작은 나무와 도로 시설 등의 변화, 그리고 빛의 세기 변화가 포함된다.

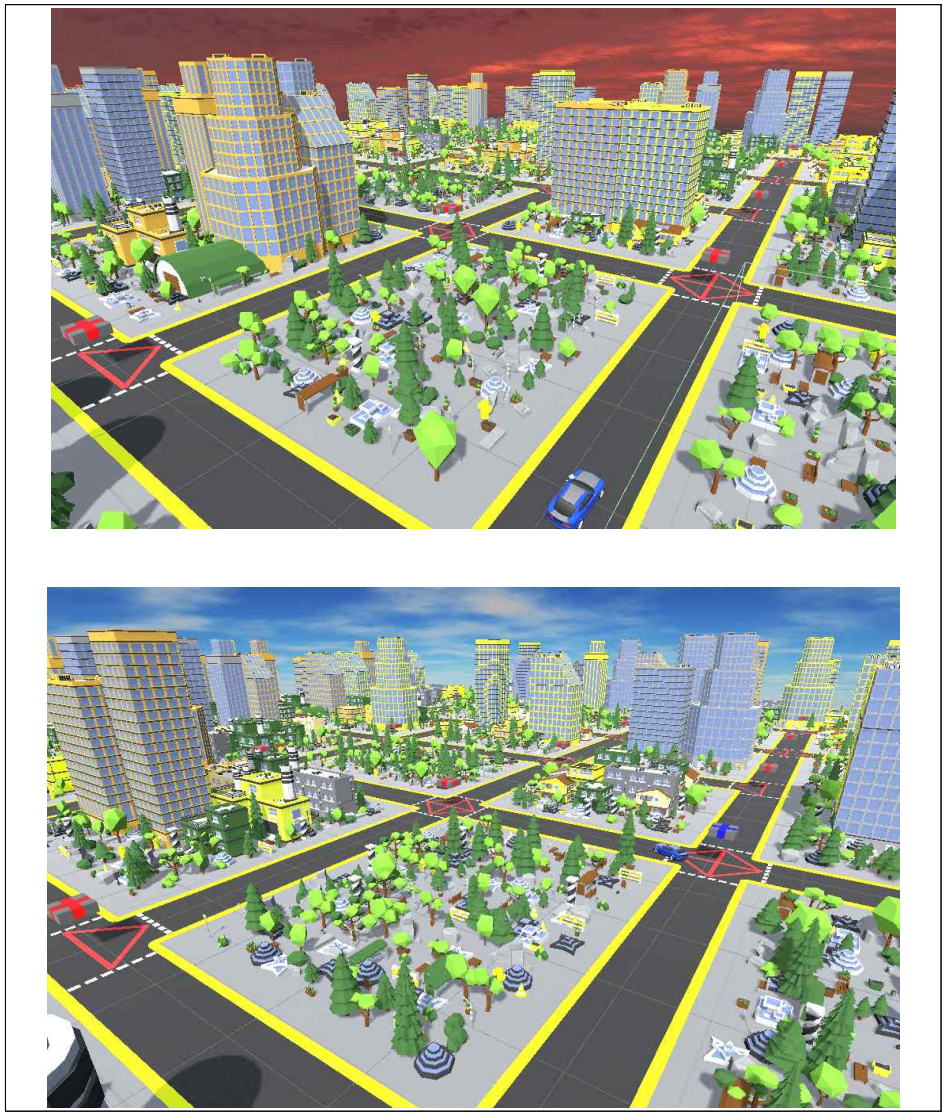

그림 17 : 주행 중 자동 및 수동으로 환경을 변화시킬 수 있다.

그림 18와 같이 수집된 데이터는 실제 카메라 데이터와 유사한 (1280*720) 고해상도 이미지 형태로 받은 뒤, Opencv를 통해 모델에 적합한 (224*224)형태로 수정되어, 10FPS의 속도로 기록되었다. 가상 공간 내의 차량은 실제 조작이 가능하며, 강화 학습 기반의 자율 주행 모델로도 운행할 수 있도록 제작하였다. 이를 활용하여 일반적인 주행 데이터를 수집할 때에는 강화 학습으로 학습된

모델을 사용하여 데이터 수집하였고, 실제 환경에서 보기 어려운 상황이나 Jetson의 기계적 제약으로 인해 발생할 수 있는 상황(도로 이탈 및 급격한 환경 변화)의 데이터는 직접 조작을 하며 수집하였다.

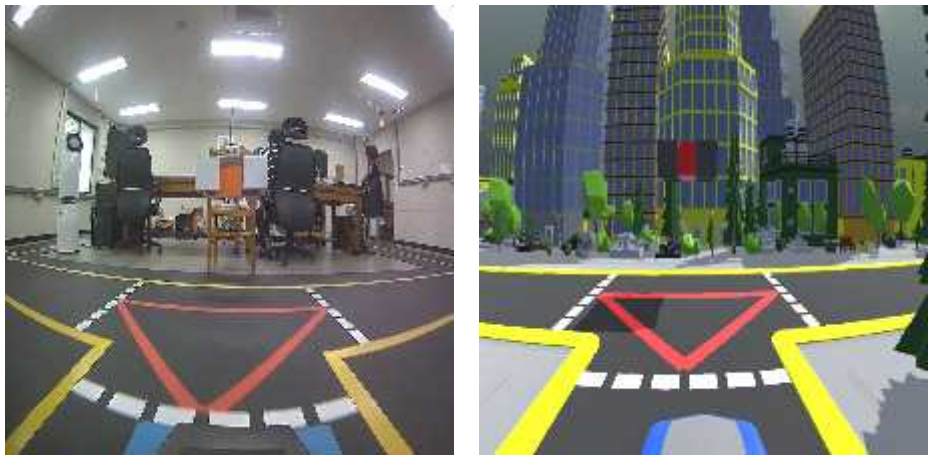

그림 18 : 좌측이 실제 공간 이미지 우측이 가상공간 이미지

3. 실험 데이터

연구를 위해 실제 공간에서 약 4시간 동안 3만장의 이미지 데이터를 수집하였다. 초기 데이터 수집 시 제한된 공간 때문에 좌, 우회전 학습이 부족하다는 문제점을 발견하였다. 이를 해결하기 위해 두 가지 방안을 검토하였다. 첫째, 좌,우회전 데이터에 더 큰 가중치를 부여한다. 둘째, RC카를 신호등과 회전 코스로 이동시켜 회전 관련 데이터를 추가 수집한다. 결과적으로 추가적인 데이터 수집이 모델의 성능 향상에 더 효과적임을 확인하였고, 이 방법으로 데이터를 추가 수집하였다.

테스트 도중 차선 이탈이 발생하여 차선 복귀에 관한 데이터 또한 추가적으로 수집하였고, 이미지 수집 과정에서 빛의 세기가 데이터 품질에 큰 영향을 줄 것으로 판단하여, 다양한 시간대와 조명 조건에서 이미지를 수집하였다. 차량 정면 이미지는 (1280*720, 10FPS)환경으로 넓은 시야각을 가져온 뒤 모델에 넣기 위해 Opencv의 resize를 이용해 (224*224, 10FPS)의 형태로 수정되어 저장하였다. 차량의 조향 상태를 기록할 때, 좌우 방향은 0.0부터 1.0까지의 연속적인 값으로 표현되었고, 속도는 1(가속) 또는 0(정지)으로 이산적인 값을 csv 파일에 저장하였다.

가상공간에서의 시각적 정보 역시 넓은 시야각을 유지하기 위해 초기 이미지 크기인 (1280x720, 10FPS)에서 resize를 통해 (224x224, 10FPS)로 조정하여 저장하였다. 더불어 차량의 조향 각도와 속도에 관한 데이터는 실제 환경에서의 데이터 수집 방식과 동일하게 처리하였다. 실제 공간에서의 데이터와 가상공간에서의 데이터는 그림 18처럼 큰 차이는 보이지 않는다.

Ⅵ. 실제 공간 자율 주행 모델

1. CNN(Convolutional Neural Network)의 구조

CNN은 이미지나 비디오 데이터와 같은 격자 형태의 고차원 데이터를 처리하는 데 특화된 딥러닝 모델이다. 기존의 인공신경망에 비해 차별화된 점은, CNN은 합성곱(Convolution) 연산을 통해 이미지 내의 패턴과 특징을 효과적으로 추출할 수 있다는 것이다. 이 합성곱 연산은 고차원 데이터의 지역적 패턴을 잡아내는 데 탁월한 역할을 한다. 전반적인 구조는 아래의 그림 19과 같이 '입력', '특징 검출', 그리고 '분류'의 3단계로 구성되어 있다.[8]

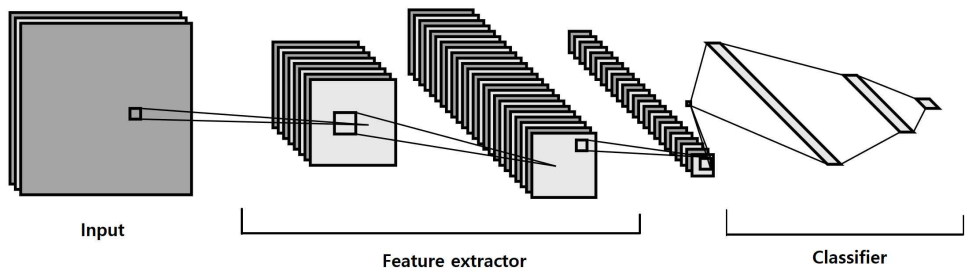

그림19 : CNN의 구조

입력 구조는 이미지의 픽셀 값을 초기 데이터로 사용한다. 특히 컬러 이미지의 경우, RGB로 이루어진 3개의 채널 값을 입력으로 받아 처리한다.

특징 검출 구조는 합성곱 계층(Convolutional Layer)과 풀링 계층(Pooling Layer)을 반복하며 이미지 내의 공간적 특징을 중심으로 인식하고 추출한다. CNN의 초기 계층들에서 주로 이러한 특징 검출 작업이 이루어지며, 계층이 깊어질수록 더 복잡하고 추상적인 특징들을 인식하게 된다. 예를 들면, 초기 계층에서는 간단한 선이나 색상 변화와 같은 기본적인 특징들이, 후반부 계층에서는 객체의 형태나 구조와 같은 고차원의 특징들이 인식된다.

분류 구조는 특징 검출 구조를 통해 추출된 정보는 분류 구조에서 활용된다. 여기서는 추출된 특징들을 바탕으로 이미지나 데이터를 특정 카테고리에 할당하는 작업이 이루어진다. 이때, 분류 구조는 비교적 간단한 구조로 평탄화 계층(Flatten Layer)과 완전 연결 계층(Fully Connected Layer), 활성화 함수(Activation Functions)로 이루어져 있다.

합성곱 계층은 그림20처럼 일정한 크기(예: 3x3)의 필터를 이용해 이미지를 슬라이딩하며 연산을 수행한다. 이 필터는 설정된 스트라이드 간격으로 이미지를 움직이면서, 해당 필터와 이미지 부분 간의 원소별 곱셈을 진행한다. 결과로 나온 출력을 '특징 맵(feature map)'이라고 부르며, 원본 이미지에서 중요한 특징들을 강조하게 된다.[9]

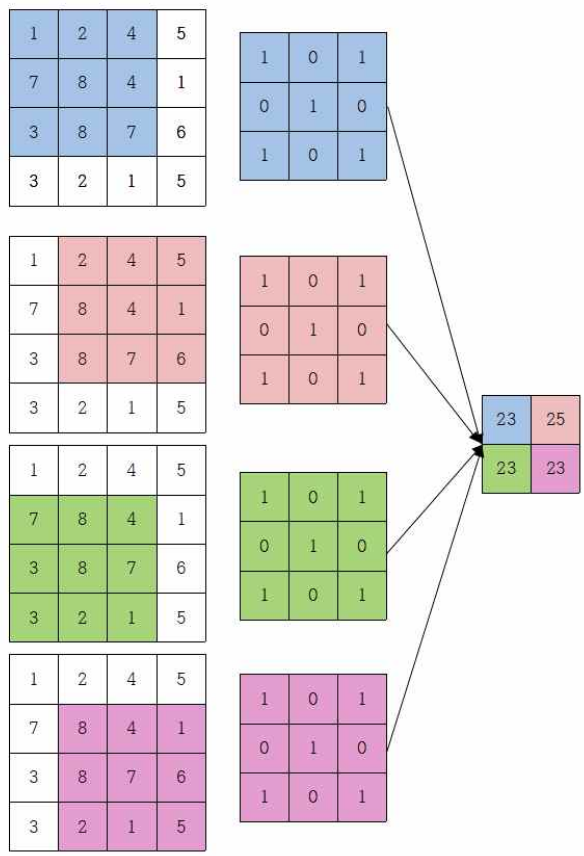

그림 20 : 합성곱 과정

풀링 계층은 이미지 처리 과정에서 중요한 역할을 한다. 특징 맵의 크기를 축소하는 동시에 이미지 내의 위치 변화에 따른 데이터의 민감한 반응을 줄이는 데에 기여한다. 이때 주로 사용되는 방법은 그림 21처럼 Max-Pooling이다. Max-Pooling은 지정된 영역 내의 값들 중에서 최댓값만을 선택하여 데이터를 압축한다. 그 외에도, Average Pooling은 해당 영역의 평균 값을 선택하며, Min Pooling은 가장 작은 값을 선택하는 방식으로 작동한다.

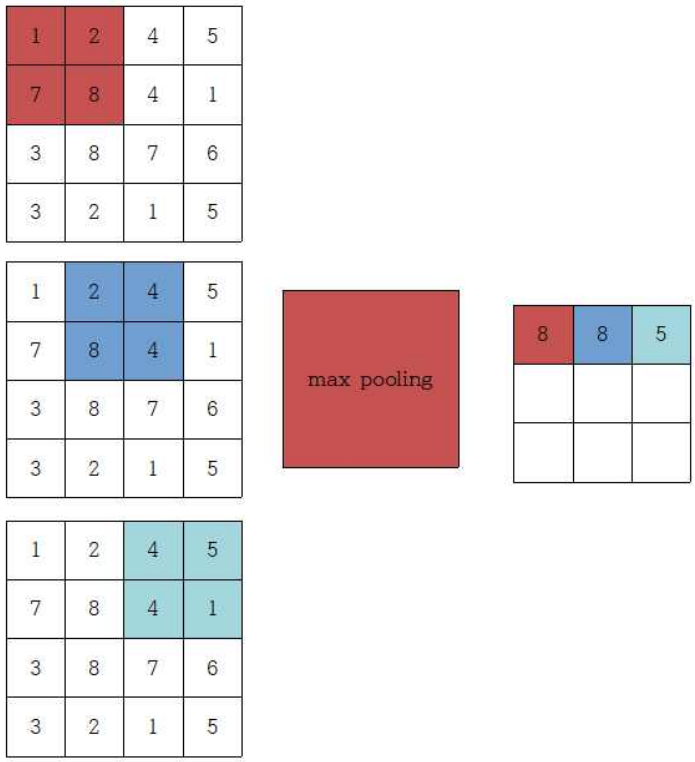

그림 21 : 풀링 과정

또한, CNN에서 이미지 처리를 위한 합성곱 연산을 진행할 때, 이미지의 크기가 줄어들거나 가장자리의 정보가 손실될 수 있는 문제가 발생할 수 있다. 이러한 문제를 방지하기 위해서 그림 22처럼 패딩(padding)이라는 기법이 사용된다. 패딩은 원본 이미지의 가장자리에 추가적인 정보를 넣어주는 작업으로, 이를 통해 이미지의 크기나 가장자리 정보를 보존할 수 있다.

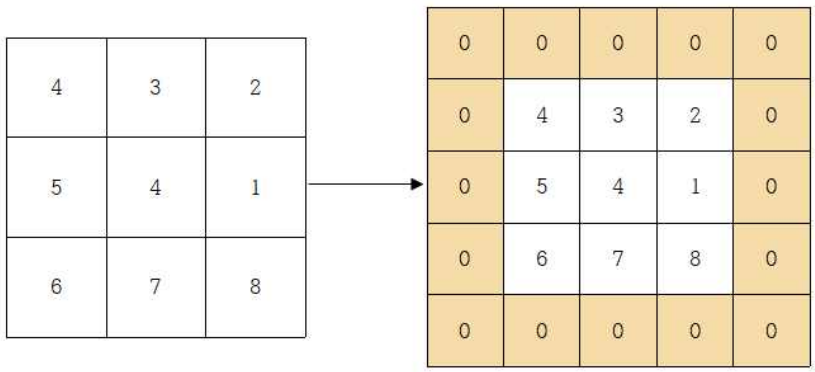

그림 22 : 패딩 과정

평탄화 계층은 그림 23의 Flatten으로 합성곱 및 풀링을 통해 얻은 2D나 3D의 특징 맵을 1D 벡터로 변환하는 역할을 하고, 변환된 1D 벡터는 후속 과정인 완전 연결 계층으로 전달된다. 완전 연결 계층은 Fully Connect로 평탄화 계층에서 출력된 1D 벡터를 입력으로 받아, 모든 뉴런이 서로 연결된 형태로 구성된다. 하나 또는 그 이상의 완전 연결 계층이 사용될 수 있으며, 각 뉴런은 이전 계층의 모든 뉴런과 상호 연결되어 있다. 이 구조를 통해 이미지의 전반적인 정보를 학습하고, 분류를 위한 특징을 도출하게 된다.

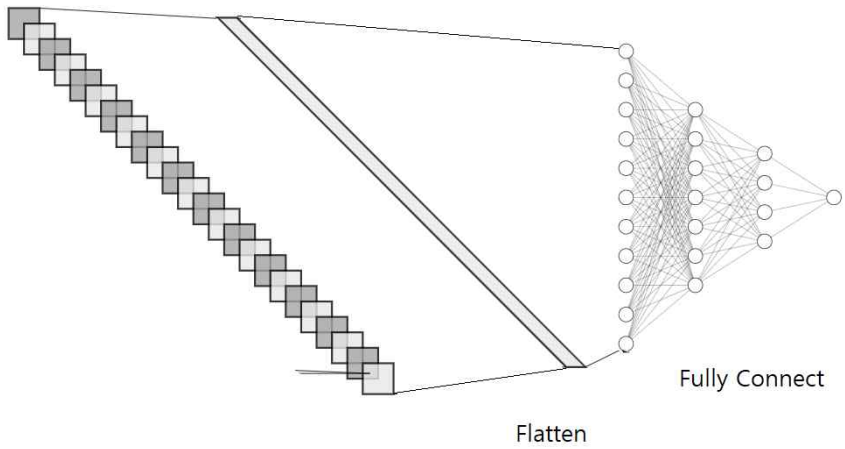

그림 23 : 분류 구조

2. 사용 모델

A. Vgg16

Vgg16은 딥러닝 분야에서 널리 사용되는 컨볼루션 뉴럴 네트워크(CNN)의 한 구조로, Visual Geometry Group(VGG)에 의해 개발되었다. 2014년 ImageNet ILSVRC(Large Scale Visual Recognition Challenge)에서 뛰어난 성능을 나타냈다. Vgg16은 그 이름에서 알 수 있듯이 그림24와 같이 16개의 레이어로 구성되어 있으며, 레이어 개수가 19개인 버전은 Vgg19로 불린다.

Vgg16의 주요 특징 중 하나는 작은 3x3 컨볼루션 필터만을 사용하여, 큰 필터의 수용 영역(receptive field)을 효과적으로 대체한다는 점이다. 예컨대, 3x3 컨볼루션을 두 번 연속적으로 사용하면, 5x5 필터와 같은 수용 영역을 얻을 수 있다. 또한 다른 중요한 특징은, 방대한 이미지넷 데이터를 통해 훈련된 후, 전이 학습(transfer learning) 기법을 통해 다양한 이미지 인식 작업에도 높은 성능을 보이고 있다는 점이다.[10]

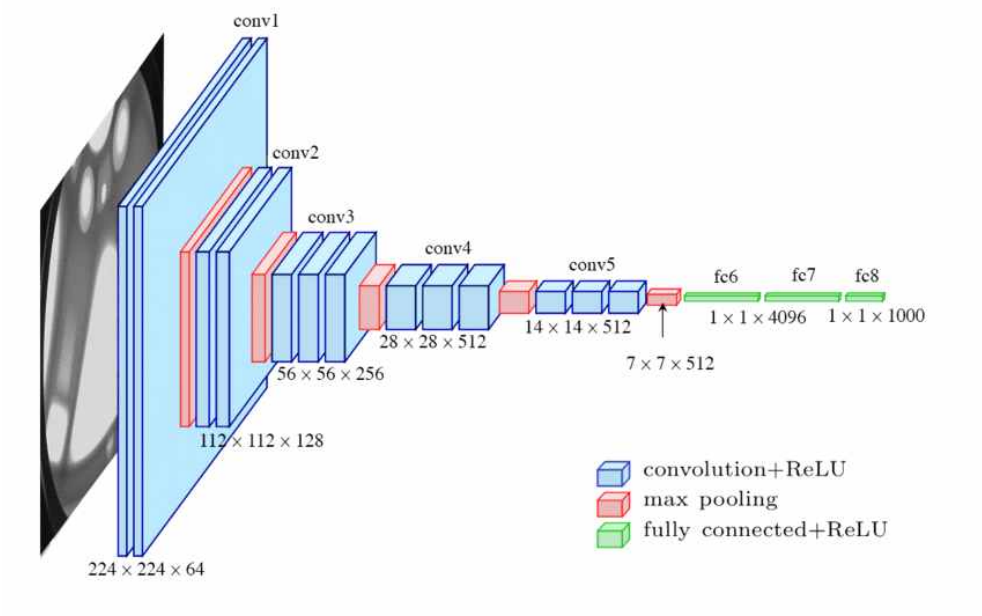

그림24 : Vgg16 구조

B. Pilot_net

Pilot_net은 엔비디아에서 제작한 자율 주행을 위한 딥 러닝 기반 네트워크로서, 실시간으로 주행 경로를 예측하는 CNN 모델을 기반으로 한다. 이 네트워크의 주요 특징은 End-to-end learning에 있다. 전통적인 자율 주행 시스템은 차선 감지, 객체 인식 등 다양한 중간 단계를 거친 후 결론을 도출하는 반면, Pilot_net은 원시 이미지로부터 운전 명령까지의 직접적인 매핑을 학습하는 end-to-end 방식을 채택한다. 모델의 구조는 그림8에서 보이는 것처럼 CNN으로, 9개의 레이어로 구성되어 있으며, 정규화 레이어, 5개의 합성곱 레이어, 그리고 3개의 완전 연결 레이어로 이루어져 있다. 이때, 대부분의 CNN 모델이 RGB 값을 기반으로 3채널 입력을 받는 것과는 다르게, Pilot_net은 YUV 평면으로 분할된 입력을 받아들인다. 추가로, 일반적으로는 사용되지 않는 5x5 필터를 사용한 점과 pooling 과정이 생략된 것도 특징이다.[6]

C. Mobile_Net_v1

Mobile_Net_v1과 Mobile_Net_v2는 모바일 기기에서 효율적으로 운용될 수 있도록 설계된 경량화된 딥러닝 모델이다. V1의 핵심 전략은 VGG 구조에서 전통적인 convolution 연산을 Depthwise separable convolution으로 대체하고 Pooling 대신 Stride를 2로 하여 Size를 축소시켜 파라미터의 개수를 획기적으로 줄이는 것이 핵심이다.

Depthwise separable convolution은 그림 25에서처럼, Depthwise Convolution과 Pointwise Convolution 두 단계의 과정으로 구성된다.

Depthwise Convolution은 입력 데이터의 각 채널에 독립적인 컨볼루션 연산을 적용한다. 이 과정에서 입력 채널의 수와 출력 채널의 수는 동일하게 유지된다. 한편, Pointwise Convolution은 출력의 채널수를 변경하는 데 사용되며, 이는 1x1 컨볼루션 연산을 통해 수행된다.

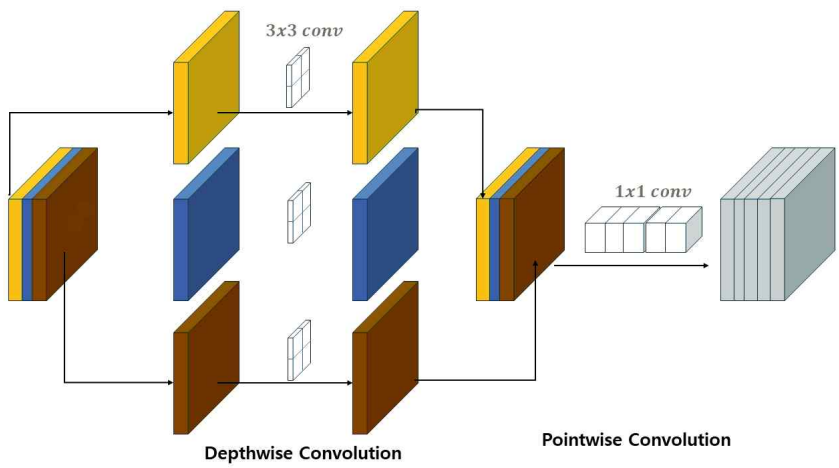

그림 25 : Depthwise Separable Convolution

실험 결과 그림 26처럼 모델 파라미터량을 줄였던 Mobile_Net_v1이 Vgg16와 비교하였을 때 성능은 크게 떨어지지 않는 결과를 보였다.[11]

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

그림 26 : Mobile_Net_v1 성능표

D. Mobile_Net_v2

Mobile_Net_v2는 그전 버전인 v1을 개선하여 개발된 모델이며, 그 아키텍처는 그림 27에 나타나 있다. 이 모델의 핵심적인 구조는 'Inverted Residual'이라는 bottleneck 구조다. 논문에서는 이 bottleneck 구조를 backbone으로 사용했을 때, Object detection 및 Segmentation 분야에서 다른 backbone 네트워크보다 더 우수한 성능을 보였다고 설명하고 있다.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

그림 27 : Mobile_net_v2 구조

모델 아키텍처에서 언급된 bottleneck 구조는 그림28에 나오는 것처럼 stride 값에 따라 두 가지 방식으로 구성되어 있다. stride가 1인 경우에는 residual network를 그대로 유지하면서 추가하였다. 반면, stride가 2인 경우에는 skip connection이 없고, convolution의 stride 값을 2로 설정하여 크기를 절반으로 축소한다. 또한, Mobile_Net_v2에서는 ReLU6라는 활성화 함수도 도입하였다. 이 함수는 기존의 ReLU 함수와 비슷하나, x 값이 0보다 클 때는 x 값을 그대로 반환하고, x 값이 6보다 클 경우에는 6을 반환하는 특성이 있다.

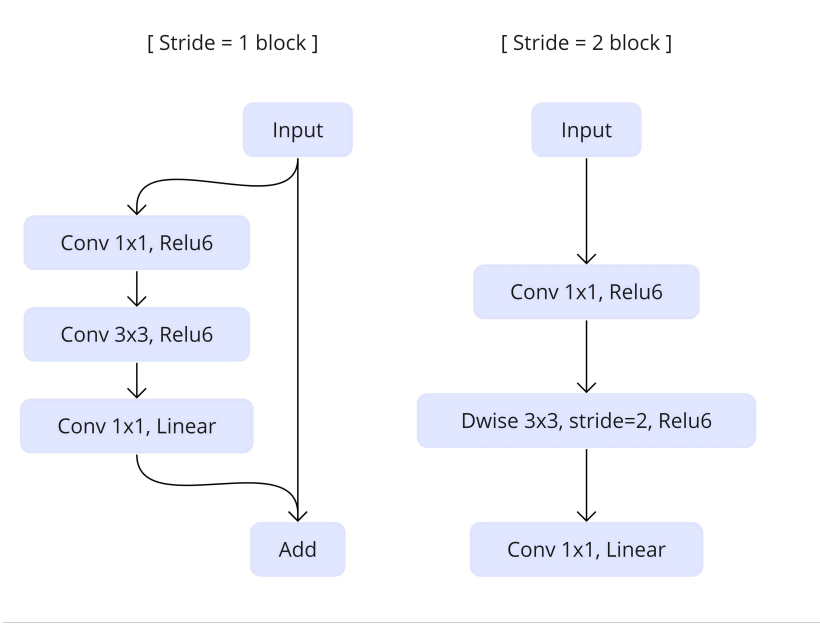

그림 28: bottleneck구조

논문 결과에 따르면 표3에서 확인할 수 있듯이, Mobile_Net_v2는 파라미터 수 및 연산 속도를 대폭 줄였음에도 불구하고 기존의 Mobile_Net_v1 및 ShuffleNet 보다 더 뛰어난 성능을 나타냈다.[12]

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

표 3 : Mobile_Net_v2 성능표

Ⅶ. 가상공간 자율 주행 모델

1. 강화학습

강화학습(Reinforcement Learning, RL)은 기계 학습의 한 분야로, 주어진 환경 안에서 에이전트가 보상을 최대화하도록 학습하는 방법론이다. 에이전트는 환경과 상호작용하며 다양한 행동을 시도하고, 결과로 받는 보상을 기반으로 학습을 진행하는데, 이는 그림 29에서 볼 수 있다. 이러한 과정을 수학적으로 모델링한 것이 'Markov Decision Process, MDP'이다.

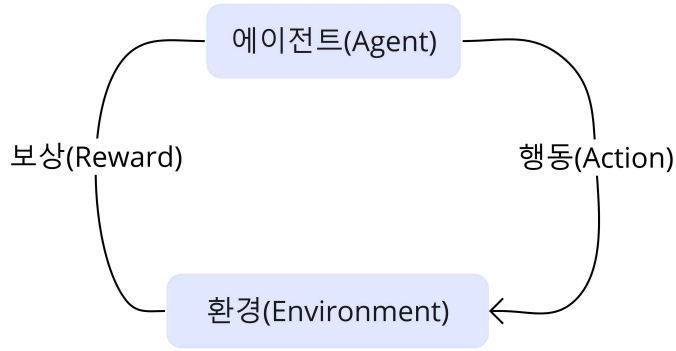

그림29 : 강화학습 구성 요소

MDP의 구성은 상태(S), 행동(A), 전이 확률(P), 그리고 보상 함수(R)로 이루어져 있다. 상태 S는 모든 가능한 환경의 상황을 나타내며, 행동 A는 에이전트가 수행할 수 있는 모든 행동의 집합이다. 전이 확률 P는 주어진 상태와 행동에 대한 다음 상태의 확률 분포를 나타내며, 수학적으로 P($s' \mid s, a$)는 상태 s에서 행동 a를 취했을 때 상태 s' 로 전이될 확률을 나타낸다. 보상 함수 R은 상태와 행동, 그리고 결과로 도달하는 다음 상태에 대한 보상을 표현한다. 보상 함수는 R(s, a, s')로 주어지며, 상태 s에서 행동 a를 선택하고 상태 s'로 전이될 때 받게 되는 보상을 의미한다.[13]

MDP의 목표는 각 상태에서 어떤 행동을 선택할지 결정하는 정책(Policy)을 찾는 것입니다. 이 정책은 수식1처럼 미래에서 받게 될 보상(Return, G)을 최대화하도록 설계되어야 한다. 리턴은 특정 시간 t에서 시작하여 미래에 받을 예상되는 보상의 할인(gamma, γ)된 합을 나타낸다.

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

수식 1 : 리턴

강화학습에서 환경은 일정하지 않고 불확실성을 가지고 있기 때문에, 동일한 상태와 행동에서도 다양한 결과가 발생할 수 있다. 이러한 불확실성을 관리하기 위해 미래의 보상에 대한 기댓값을 계산하며, 이를 통해 평균적인 효과를 기대한다. 이때 중요한 개념이 가치함수(Value Function)로, 이는 미래의 보상에 대한 기댓값을 나타낸다. 가치함수는 상태 가치함수(V)와 행동 가치함수(Q)로 구분되며, 특정 강화학습 문제나 알고리즘의 특성에 따라 적절한 가치 함수를 선택하여 정책을 구축한다.

상태 가치 함수(State Value Function, V(s))는 특정 상태 s에서 시작하여 예상되는 미래의 누적 보상을 나타낸다. 이는 수식2에서 수학적으로 확인할 수 있다.

$$v(s) = E[G_t | S_t = s]$$

수식2 : 상태 가치 함수

행동 가치함수(Action Value Function, Q(s,a))는 상태 s에서 행동 a를 선택했을 때 예상되는 미래의 누적 보상을 나타낸다. 이 함수는 수식3에서 수학적으로 표현되어 있다.

$$q_\pi(s,a) = E_\pi[G_t | S_t = s, A_t = a]$$

수식3 : 행동 가치 함수

2. DQN

행동 가치를 사용하는 알고리즘 중 DQN(Deep Q-Network)은 강화학습에서 Q-Learning알고리즘과 심층 심경함(DNN)을 결합한 방법으로 행동 가치 함수를 상태를 넣어 Q값을 출력하는 DNN을 통해 구한다. 이후 환경에 맞게 MC(Monte Carlo)와 TD(Temporal Difference)를 선택하여 환경에서 직접 돌아보고 값을 가져온다. DNN으로 구한 행동 가치 함수 값과 오차 제곱을 통해 Loss를 구하고 DNN을 업데이트한다.[14]

MC방법은 그림 30처럼 에피소드를 완전히 진행한 후에 학습을 진행할 수 있다. 즉, 에이전트가 시작 상태에서 목표 상태에 도달할 때까지의 경로를 완전히 관찰한 후에 그 경로에서 얻은 보상들을 사용하여 가치 함수(value function)을 업데이트한다.

TD방법은 완전히 진행이 끝난 에피소드만 학습할 수 있는 MC의 문제점을 해결하고자 그림 31처럼 한 단계나 몇 단계만 진행한 후에 학습을 진행한다.

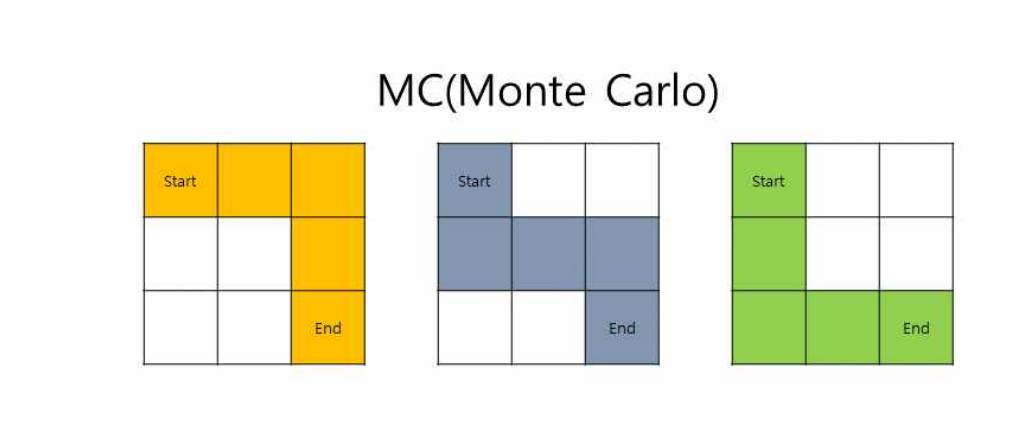

그림 30 : Monte Carlo 방식

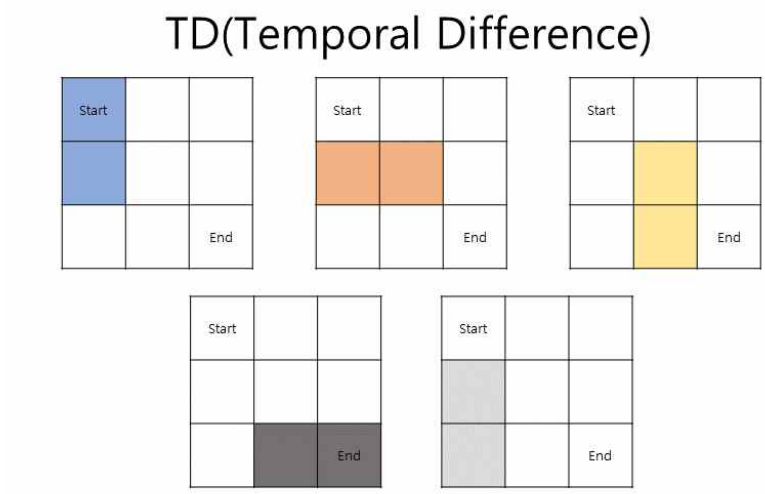

그림 31 : Temporal Difference 방식

3. PPO 알고리즘

PPO 알고리즘은 TRPO 알고리즘의 복잡한 계산 과정을 단순화한 버전으로, 안정적인 학습을 위한 제약 조건을 보다 간결한 목적 함수로 구현하였다. TRPO는 정책 변경의 극단적인 변동을 방지하기 위한 복잡한 최적화 문제에 대한 해결책을 제시하지만, PPO는 클리핑이나 적응적 KL 페널티 같은 방법들을 도입함으로써 문제를 단순화하였다.

TRPO는 기존에 샘플 데이터를 한 번 사용한 후 버리는 방식을 Importance Sampling 방법을 도입함으로써 데이터를 여러 번 재활용할 수 있게 만든 알고리즘이다. 전통적인 Policy Gradient 방식은 수식4와 같이 LOSS를 정의하여 수식5처럼 기울기를 계산하고, 이를 바탕으로 정책을 업데이트하는 방식을 사용하여 업데이트를 진행한다. [15]

$$L^{PG}(\theta) = \hat{E}_t[\log \pi_\theta(a_t|s_t)\hat{A}_t]$$

수식4 : Policy Gradient 목적 함수

$$\hat{g} = \hat{E}_t[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t]$$

수식5 : Policy Gradient 기울기

파라미터가 수정되면서 이전 데이터를 재사용하는 것은 불가능했지만 TRPO에서는 수식6처럼 '파라미터 간의 차이가 작을 때'라는 특정 조건을 넣음으로써 수식7처럼 기울기식을 수정할 수 있었고, 이를 바탕으로 수식8처럼 목적함수를 수정하여 과거 데이터의 재사용을 가능하게 했다.

결과적으로 TRPO는 수식9이 최대가 되도록 학습하게 된다.

$$\hat{E}_t[KL[\pi_{\theta_{old}}(\bullet|s_t), \pi_\theta(\bullet|s_t)]] \leq \delta$$

수식6 : 제약조건

$$\nabla_\theta \log f(\theta)|_{\theta_{old}} = \frac{\nabla_\theta f(\theta)|_{\theta_{old}}}{f(\theta_{old})} = \nabla_\theta \left(\frac{f(\theta)}{f(\theta_{old})} \right)|_{\theta_{old}}$$

수식7 : 기울기 수식의 변화

$$L^{IS}_{\theta_{old}}(\theta) = \hat{E}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t]$$

수식8 : TRPO의 목표함수

$$\hat{E}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t] - \beta \hat{E}_t[KL[\pi_{\theta_{old}}(\bullet|s_t), \pi_\theta(\bullet|s_t)]]$$

수식9 : TRPO의 목적함수 값이 최대가 되는 방향으로 학습하게 된다.

PPO 알고리즘 수식10을 TRPO수식에 대입하고 11처럼 클리핑 방식으로 간소화함으로써, 복잡한 최적화 문제를 효과적으로 회피하며 연산 부담도 크게 줄였다. 이 클리핑된 목적 함수는 극단적인 정책 변화를 방지하면서도 효율적인 학습을 보장한다. 그 결과, PPO는 다양한 강화학습 문제에서 안정성과 함께 높은 성능을 보이는 것으로 널리 인정받고 있다.

$$r_t(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}, r_t(\theta_{old}) = 1$$

수식10 : 목적함수를 간단하게 위한 수식

$$r^{clip}(\theta) = \hat{E}_t[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t]$$

수식11 : PPO알고리즘의 목적 함수

4. 가상공간 자율 주행 모델 제작 및 학습

가상 공간에서 주행을 담당하는 모델은 차량의 전방 이미지를 활용하지 않고 라이다 센서를 사용하였다. 모델의 학습 방식으로는 일반적인 지도학습이 아닌 PPO 알고리즘 기반의 강화학습을 선택하였다.

전방 이미지를 사용하지 않은 이유는 해당 이미지가 너무 많은 정보를 함축하고 있어, 모델 학습에 어려움을 줄 수 있다고 판단되었기 때문이다. 대신, 그림 32과 같이 더 직관적인 라이다 센서를 활용하였다.

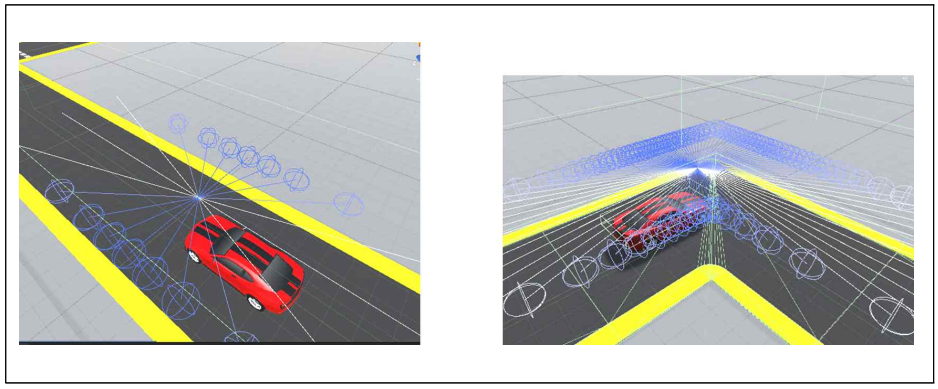

그림 32 : 차량의 라이다 센서

강화학습을 선택한 이유는 가상 환경이 강화학습의 데이터 요구량 문제를 해결할 수 있을 뿐만 아니라, 다양한 시나리오에서의 주행 전략 학습에도 효과적이라고 판단되었기 때문이다. 강화학습은 복잡한 환경에서 다양한 상황을 겪으며 최적의 전략을 찾아내는 학습 방법으로, 이를 가상공간에서 차량 주행 학습에 적용하기 적절하다고 생각되었다. 보상 체계는 표4처럼 설정하여 학습을 진행하였다.

종류	보상(프레임 당)
도로에서 벗어나지 않고 생존	1.0
이전 조향각도와 현재 조향각도의 차이가 작으면	0.001
도로를 벗어나면	-500

표 4 : 보상 체계

학습된 차량은 제작한 가상공간에서 100%사고 없이 최적의 회전 각도로 회전하는 것을 확인하였고, 이로 인해 자동으로 다양한 데이터를 수집할 수 있었다.

Ⅷ. 실험 및 결과 분석

1. 실험 방법

본 논문은 가상공간에서 수집한 데이터의 성능 평가를 목적으로 여러 실험을 진행하였다.

첫 번째 실험에서는 세 가지 모델(Vgg16, Mobile_Net_v2, Pilot_Net)과 두 종류의 데이터(실제 데이터, 합성 데이터)를 사용하였다. 준비된 데이터 셋은 환경에서 수집된 차량의 정면 이미지 30,000장으로 구성되어 있으며, 각 이미지에는 해당 차량의 조향 각도 정보만이 포함되어 있다. 따라서 모델은 차량의 정면 이미지를 바탕으로 조향 각도를 예측하는 작업을 수행하게 된다. 성능 평가는 Mean Squared Error(MSE) 오차를 기준으로 하였으며, 이를 위해 테스트 데이터로 15,000장의 이미지를 사용하였다. (표 5)

실험 모델		
Vgg16	Pilot_Net	Mobile_Net_v2
학습 데이터(정면 이미지, 조향각도)		
실제 데이터(3만장)	합성 데이터(6만장)	

표5 : 모델과 데이터

두 번째 실험에서는 첫 번째 실험에서 우수한 성능을 보인 Mobile_Net_v2 모델을 사용하였고, 데이터에는 차량의 정면 이미지, 조향 각도, 그리고 가속 여부가 포함되었다. 이번 실험에서는 모델이 차량의 조향 각도뿐만 아니라 가속 여부까지 예측하도록 수정하였다. 성능 평가는 조향 각도의 Mean Squared Error(MSE)와 차량 가속의 CrossEntropy를 합한 값을 Loss로 사용하였다. MSE와 CrossEntropy 값 사이에 큰 차이가 있어 학습에 문제가 발생하여, CrossEntropy 값에 0.01의 배율을 적용하여 Loss를 조정하였다. 사용된 학습 데이터는 가상 데이터, 실제 데이터, 그리고 두 데이터를 혼합한 형태로 구성되어 있으며, 자세

한 내용은 아래 표 6에서 확인할 수 있다.

	가상데이터	실제데이터	혼합데이터
개수	30,000장	30,000장	가상데이터 + 실제 데이터

표6 : 학습 데이터 종류

세 번째 실험에서는 두 번째 실험을 통해 개발된 두 모델(실제 데이터만을 이용하여 학습한 모델과 혼합 데이터를 이용하여 학습한 모델)을 이용하여 실제 환경에서 3회의 주행 테스트를 실시하였다. 각 주행은 실시간으로 진행되었고, 평가는 주행 중 발생할 수 있는 사고를 기반으로 점수를 매기는 방식을 사용하였으며, 이에 대한 구체적인 지표는 아래 표 7에서 확인할 수 있다.

	도로 이탈	신호 무시	도로 침범
점수	-2점	-2점	-1점

표7 : 학습 데이터 종류

2. 실험 결과

첫 번째 실험 결과에 따르면, 표 8에서 볼 수 있듯이 Mobile_Net_v2와 Pilot_Net은 합성 데이터를 사용할 때 5%~30%성능이 향상되었다. Vgg16 모델은 데이터의 종류에 상관없이 뛰어난 성능을 보였지만, Jetson에서 실시간으로 예측을 실행하는 데 있어서는 일정한 제약이 있었다. 이러한 문제를 해결하기 위해, Vgg16과 성능이 비슷하면서도 실시간 예측에 더 적합한 Mobile_Net_v2를 선택하여 두 번째 실험을 수행하였다.

	Mobile_Net_v2		Pilot_Net		Vgg16	
	합성 데이터	실제 데이터	합성 데이터	실제 데이터	합성 데이터	실제 데이터
test (MSE)	0.67	0.71	1.99	2.87	0.58	0.58

표8 : 모델 성능 표(angle만 학습)

두 번째 실험에서는 실시간 예측의 효율성을 위해 Vgg16 대신 Mobile_Net_v2를 사용하여 진행하였고, 그 결과는 표 9에 기록되어 있다. 가상 데이터만으로 학습한 경우 실제 데이터를 사용한 것보다 성능이 다소 떨어졌다. 그러나 가상 데이터와 실제 데이터를 결합한 합성 데이터를 사용하면, 실제 데이터만을 사용한 경우보다 1% 높은 성능을 보여주었다.

Mobile_Net_v2		실제 데이터	가상 데이터	합성 데이터
성능 (LOSS)	Total	1.24	11.9	1.23
	Speed	0.35	0.39	0.34
	Angle	0.88	11.5	0.88

표9 : 데이터 별 성능 표(speed + angle동시 학습)

세 번째 실험 결과는 표10에서 볼 수 있다. Mobile_Net_v2 모델을 사용하여 실제 데이터로 학습시킨 모델은 도로 침범의 수치가 합성 데이터보다 많았으며, 합성 데이터로 학습시킨 모델은 도로 이탈의 문제가 많이 발생하였다. 하지만 총 감점 점수는 같은 모습을 보여주었다.

Mobile_Net_v2		실제 데이터	합성 데이터
감점	도로 이탈	5회	8회
	신호 무시	3회	3회
	도로 침범	9회	3회
	총 감점	25점	25점

표 10 : 실제 주행 결과

3. 결과 분석 및 결론

첫 번째 실험에서 Vgg16은 높은 성능을 보였지만, 1억 개의 파라미터 때문에 실시간 예측에는 한계가 있었다. 반면, Mobile_Net_v2는 Depthwise Separable Convolution, Inverted Residual Blocks, 그리고 Skip Connections와 같은 기술을 적용하여, 단 3백만 개의 파라미터로 Vgg16과 유사한 성능을 달성했다. 이 결과로 인해 자원이 제한된 기기(예: 휴대폰, 일반 전자 기기)에서 높은 성능이 필요할 때 Mobile_Net_v2의 활용이 효과적임을 확인했다. 반면, Pilot_Net은 학습 파라미터가 150만개로 적어 실시간 예측에 유용했으나, Mobile_Net_v2처럼 성능 향상에 기여하는 특별한 구조가 없어 낮은 성능의 원인으로 보인다.

첫 번째와 두 번째 실험 결과, 가상 데이터로만 학습된 모델의 성능은 실제 데이터로만 학습된 모델에 비해 상대적으로 떨어졌다. 이러한 성능 차이는 가상 환경과 실제 환경 간의 물리적 특성 차이(회전 반경), 가상 데이터가 현실의 복잡한 다양성을 완전히 반영하지 못하는 점에서 기인하는 것으로 판단된다. 그럼에도 불구하고, 합성 데이터와 실제 데이터를 조합하여 사용한 실험에서는 모델의 성능이 개선되었다는 점이 확인되었다. 이러한 결과는 가상 환경의 현실감이 개선됨에 따라 학습 모델의 성능도 상응하여 향상될 가능성을 시사한다.

두 번째와 세 번째 실험을 통해 합성 데이터의 활용이 모델의 수치적 성능을 향상시키는데 기여한 것이 확인되었다. 그러나 이러한 성능 향상이 실제 주행 환경에서는 분명하게 나타나지 않았다.

본 연구에서는 가상 데이터를 활용하여 모델의 성능을 향상시키려 하였다. 실험 결과, 기이한 데이터임에도 불구하고 세 가지 모델 중 두 모델에서 실제로 성능 향상이 관찰되었다. 이 결과로부터 가상 데이터 활용이 모델 성능 향상에 긍정적인 영향을 미칠 수 있음이 확인되었다. 그러나 모델이 실제 주행 환경에서 더 나은 성능을 보이기 위해서는, 현실 환경의 특성과 조건을 최대한 반영한 가상 데이터를 생성하는 것이 필요하다는 점도 뚜렷이 드러났다.

Ⅸ. 인사말

학부 시절부터 제 여러 부족한 점에도 불구하고 저를 지도해 주신 이형원 교수님께 진심으로 감사의 말씀을 드립니다. 교수님은 일반적인 방법보다는 저에게 더 큰 도움이 되는 지도를 해 주셨고, 특히 인공지능을 수학의 관점에서 바라보며 귀중한 조언을 주셨습니다. 그 덕분에 많은 궁금증을 해결할 수 있었습니다. 자율주행 연구 과정에서도 교수님의 지속적인 지원과 조언 덕분에 많은 것을 배우며 성장할 수 있었습니다. 이런 기회를 주신 교수님께 다시 한 번 깊이 감사드립니다.

또한, 인공지능에 대한 호기심을 자극해주시고 이 분야의 길을 처음 알려주신 김경이 교수님께도 진심으로 감사드립니다. 교수님께서는 인공지능의 다양한 분야를 제게 소개해주시며, 하나의 주제에만 국한되지 않고 여러 가능성을 보여주셨습니다. 이로 인해 많은 것을 경험하고, 자율주행 연구를 무사히 마칠 수 있었습니다. 교수님께 다시 한 번 깊이 감사의 말씀을 드립니다.

마지막으로 언제나 나의 넓은 그늘이었던 부모님께 깊은 감사를 드립니다. 부모님의 끊임없는 격려와 지지가 있었기에 많은 어려움을 극복하고 이지점에 도달할 수 있었습니다. 언제나 저의 편이 되어 주시고, 저의 꿈을 함께 해주신 부모님께 얼마나 감사한지 말로 다 표현할 수 없습니다. 정말로 감사합니다.

Ⅹ. 참고 문헌

[1] 안일구, 배광호, 이시우. “콘볼루션 신경망 (CNN)과 다양한 이미지 증강기법을 이용한 혀 영역 분할“ 의공학회지 42, no.5 (2021) : 201-210.

[2] 최영원, 이영우, 채홍석. “CNN 모델 평가를 위한 이미지 데이터 증강 도구 개발,“ 소프트웨어공학 소사이어티 논문지, vol. 29, no. 1 (2022) : 13-21.

[3] Adrien Gaidon, Qiao Wang, Yohann Cabon, et al. Virtual Worlds as Proxy for Multi-Object Tracking Analysis, 2016, arXiv:1605.06457.

[4] Jonathan Tremblay, Aayush Prakash, David Acuna, et al. Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization, 2018, arXiv:1804.06516.

[5] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, et al. Driving in the Matrix: Can Virtual Worlds Replace Human-Generated Annotations for Real World Tasks?, 2016, arXiv:1610.01983.

[6] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, et al. End to End Learning for Self-Driving Cars, 2016, arXiv:1604.07316.

[7] 유승준, 유상곤, 황성수. PilotNet을 이용한 모형 차량(HENES-T870)의 자율주행 시뮬레이션, 한국정보과학회 2022 한국컴퓨터종합학술대회, VOL 49 NO. 01(2022) PP. 2210 ~ 2212

[8] Jie Wang, Zihao Li. Research on Face Recognition Based on CNN, IOP Conference Series: Earth and Environmental Science 170, no.3 (2018) : 032110.

[9] Lingxi Xie, Alan Yuille. Genetic CNN. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, 1379:1388

[10] Min-ji Byon, Eun-joo Jun, Ji-soo Kim, et al. Evaluation of VGG-16 deep learning algorithm for dental caries classification, Journal of Korean Academy of Oral Health 45(4), 2021: 227-232.

[11] Andrew G. Howard, Menglong Zhu, Bo Chen, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 2017, arXiv:1704.04861

[12] Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, 4510:4520.

[13] 이승준, 장병탁. 복잡계의 위상특성을 이용한 MDP 학습의 효율 분석. 한국정보과학회 학술발표논문집,33(1), 2006, 232-234.

[14] 박광석, 박진만, 윤완규, et al. 스마트 팩토리 구축을 위한 동적인 환경에서의 DQN 강화학습 기반 로봇 최적 이동 경로 탐색. 한국통신학회논문지, 2019, 44.12: 2269-2279.

[15] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, et al. Implementation matters in deep policy gradients: A case study on ppo and trpo, 2020, arXiv:2005.12729