AirSim 기반 자율 주행을 위한 심층 강화 학습

Title	Deep reinforcement learning for autonomous driving in AirSim environment
Abstract	본 문서는 AirSim 시뮬레이터 환경에서 자율 주행을 위한 심층 강화 학습을 연구한 논문이다.

제출일 : 2021 년 7 월 17 일
지도교수 : 김 세 화 교수님

정보통신공학과	201600055	강 승 환
정보통신공학과	201800528	김 가 영
정보통신공학과	201801841	서 우 빈

제목 : AirSim 기반 자율 주행을 위한 심층 강화학습

대 학 : 공 과 대 학 학 과 : 정보통신공학과

학 번 : 201600055 성 명 : 강 승 환
201800528 김 가 영
201801841 서 우 빈

이 논문을 제출 하오니 승인하여 주십시오.

2021 년 7 월 27 일

성 명 : 강 승 환 (인)

김 가 영 (인)

서 우 빈 (인)

위 학생의 논문 제출을 승인함.

2021 년 7 월 17 일

지 도 교 수 : 김 세 화 (인)

요 약

본 논문에서는 AirSim 시뮬레이터 환경에서 자율 주행을 위한 심층 강화 학습을 연구한다. 기존 연구인 'The Autonomous Driving Cookbook'에서 제안한 모델을 기반으로 해당 기존 모델의 성능을 향상시키기 위해 두 가지 모델을 제안한다. 이때, 기존 연구의 한계점을 극복하기 위한 새로운 신경망 구조와 보상 함수를 제안한다. 또한 제안한 두 가지 제안 모델들과 기존 모델의 성능 평가를 진행하여 제안 모델들의 성능과 한계점을 파악한다. 이를 통해 자율 주행에 영향을 미치는 새로운 신경망 구조 및 심층 강화 학습 모델을 제안하는 것을 목적으로 한다.

(제안 모델 1)

기존 모델의 한계를 극복하기 위해 처음 제안한 모델로, 기존 모델의 전체적인 신경망 구조를 변경한다. 신경망의 입력층에서는 차량 전면 이미지 이외의 새로운 정보를 입력으로 추가한다. 또한 성능 향상을 위한 새로운 보상 함수를 제안한다.

(제안 모델 2)

제안 모델 2를 이용하여 제안 모델 1의 한계점을 극복하고자 한다. 속도와 연관을 갖는 새로운 보상 값을 차량 제어에 있어 필수적인 속도 제어의 보상 값으로 사용한다. 기본적인 steering 제어는 제안 모델 1과 동일한 보상 값을 사용한다.

(성능 평가 분석)

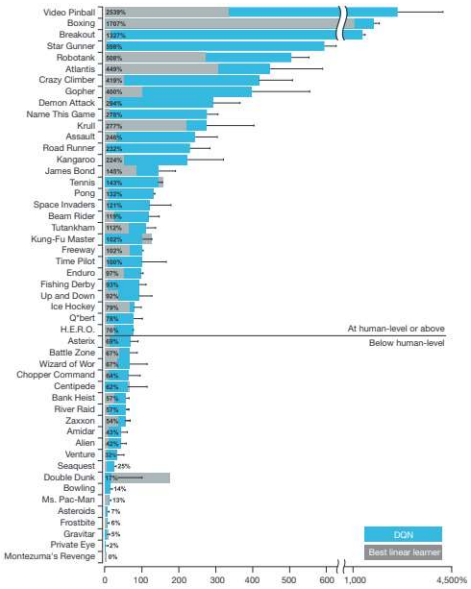
제안 모델 1과 기존 모델의 성능 평가를 진행하였다. 성능 평가는 훈련 성능과 테스트 성능에 대해 진행하였다. 훈련 성능과 테스트 성능은 주행 시간에 대해 측정하였다. 훈련 성능은 같은 시간 훈련 동안 가장 긴 주행 시간으로 비교하였으며, 제안 모델 1이 기존 모델에 비해 273% 더 향상된 성능을 가졌다. 테스트는 같은 시간 동안 훈련시킨 모델을 사용하여 진행하였으며 이때 테스트 성능 또한 기존 모델에 비해 제안 모델 1이 310% 향상된 성능을 가졌다.

목차

그림 목차

1 서론

1.1 주제 선정 이유



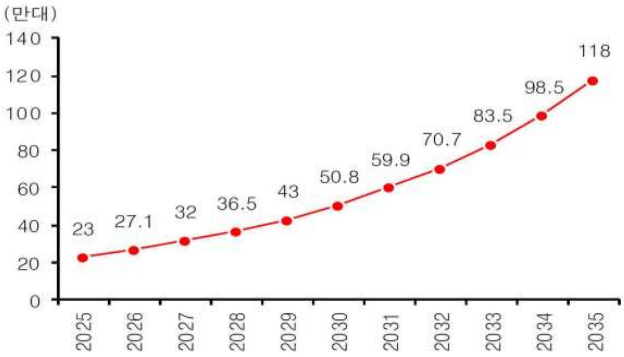


Figure 1 심층 강화 학습을 사용한 게임 성능 비교[1] Figure 2 완전자율주행차 세계 보급률 전망[2]

2016 년 인공지능 바둑 프로그램 알파고가 이세돌을 상대로 4:1 승리를 따냈다. 심층 강화 학습은 기계 학습의 한 종류인 강화 학습과 심층 신경망을 결합한 것으로, 알파고는 심층 강화 학습으로 학습시킨 인공지능이다. 알파고의 등장으로 기계 학습 훈련 방식으로써 심층 강화 학습에 대한 관심도가 높아졌다. 이후 알파고 제로[3]와 뮤제로[4] 등 심층 강화 학습을 이용한 알파고의 후속 연구들이 등장하였으며 이와 같은 시뮬레이션 가능한 게임 분야에 있어 심층 강화 학습이 매우 좋은 성과를 거두었다. Figure 1 에서는 심층 강화 학습(청색 막대)이 선형 학습(회색 막대)에 비해 대부분 뛰어나며, 대부분의 게임에서 인간보다 더 뛰어난 결과를 얻는 것을 보여준다.

인공지능을 통한 자율 주행은 현재 중요한 연구 과제이다. 5G 네트워크의 도입으로 인하여, 자율 주행 시대로의 진입이 가속화되었고, 이미 많은 자율 주행 자동차들이 개발되었으며 개발 중에 있다. 이러한 자율 주행 자동차는 점차 세상에 보급될 것이고, Figure 2 에서는 자율 주행 자동차의 보급이 지속적으로 늘어날 것임을 보여준다. 이와 같이 자율 주행 자동차의 성능과 안정성은 매우 중요한 문제이다.

1.2 본 연구의 목표

Figure 1 과 같이 게임과 같은 시뮬레이션이 가능한 환경에서의 심층 강화 학습의 효과는 검증되었다. 자율 주행을 위한 자동차 시뮬레이션 환경을 이용하여 심층 강화 학습을 통해 자율 주행에서 효과를 거두는 것은 자율 주행 자동차에 대한 관심이 높아지는 시점에서 집중할 만한 연구 과제라고 생각된다.

본 논문에서는 자율 주행 자동차를 위한 심층 강화 학습 방법에 대한 연구를 진행하는 것을 목표로 한다. [5]에서 개발한 기존 모델을 기반으로 해당 모델의 한계점을 극복한 새로운 모델들을 제안함으로써 자율 주행에서의 성능 향상을 목표로 한다.

2 관련 기술 동향 조사 및 분석

2.1 기반기술

2.1.1 AirSim

Figure 3 AirSim 시뮬레이터의 모습

AirSim 은 Microsoft 에서 배포한 오픈소스 자동차 시뮬레이터로, Unreal Engine 기반이다. AirSim 시뮬레이터는 간편한 환경 구성과 실시간 하드웨어 컨트롤을 지원하고, API 를 제공한다. AirSim 시뮬레이터의 경쟁 시뮬레이터로는 여러 시뮬레이터가 있으며, 그중 Unreal Engine 기반인 Carla[6]와 Deepdrive[7]가 있다. Carla 는 CVC (Computer Vision Center)에서 배포한 오픈소스 자동차 시뮬레이터로, 환경 설정이 쉽지만 실시간 하드웨어 컨트롤을 지원하지 않는다[6]. Deepdrive 는 Parallel Domain 에서 배포한 오픈소스 자동차 시뮬레이터로, 높은 FPS 를 제공하지만 불충분한 documentation 과 실시간 하드웨어 컨트롤을 지원하지 않는다[7]. 따라서 본 논문에서는 이러한 단점으로 인하여 AirSim 시뮬레이터를 선택하였다.

2.1.2 CNN (Convolutional Neural Network)

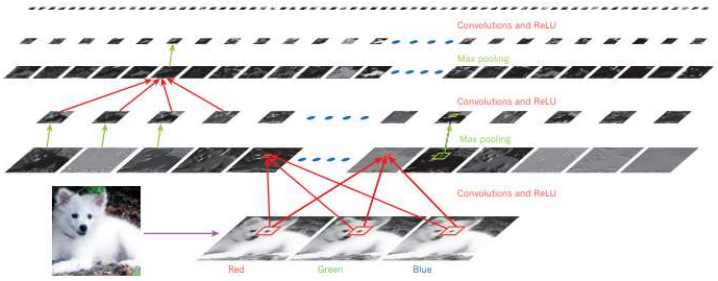


Figure 4 CNN 내부[8]

CNN[8]은 합성곱 신경망으로, 시각적 영상을 분석하는 데 사용되는 다층의 인공신경망의 한 종류이다. Figure 4 는 CNN 내부 구조를 보여주는 그림으로, CNN 이 내부에서 어떻게 동작하는지를 보여준다. 심층 학습에서 심층 신경망으로 분류되며, 시각적 영상 분석에 주로 사용된다. 본 논문에서는 CNN 을 이용하여 전반적인 모델의 입력을 처리한다.

2.1.3 강화 학습

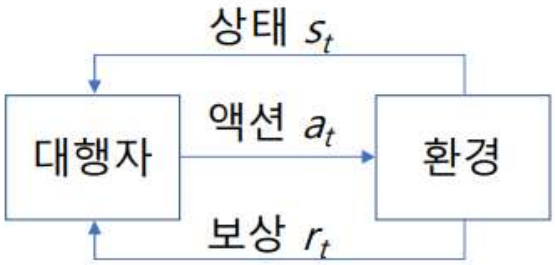


Figure 5 강화 학습의 기본 구성 요소[9]

Figure 5 는 강화 학습의 기본 구성 요소를 그림으로 표현한 것이다. 그림에서 대행자(agent)는 주어진 문제 상황에서 행동하는 주체이며, 환경(environment)은 주어진 문제의 환경을 의미한다. 상태(state)는 현재 시점에서의 상황을 나타내는 값이며 대행자가 환경으로부터 상태를 받아 대행자가 취할 수 있는 선택지인 액션(action)을 행한다. 액션을 행하면 환경은 변하게 되고, 이에 따른 보상 값인 보상(reward)을 받는다. 여기서 정책(policy)은 대행자가 판단하는 방식을 의미하며, 가치(value)는 현재의 보상 값에 미래의 보상 값을 더한 값을 의미한다.

본 논문에서는 강화 학습의 한 종류인 Q-learning 을 사용한다. Q-learning 은 현재 상태에서 주어진 액션을 수행하는 것이 가져올 Q-Value 값을 측정하는 함수인 Q 함수를 학습하며 최적의 정책을 학습하는 방법이다. Q-value 는 현재의 상태에서 보상 값과 미래의 보상 값에 대한 누적합을 의미하고, Q-value 를 최대로 하는 Q 함수는 다음과 같다.

$$V^*(s) = \max_a \mathop{\mathrm{E}}_{s' \sim P}[r(s,a) + \gamma V^*(s')]$$

$$Q^*(s,a) = \mathop{\mathrm{E}}_{s' \sim P}\left[r(s,a) + \gamma \max_{a'} Q^*(s',a')\right]$$

Bellman 방정식

위 수식에서 앞으로 주어지는 보상이 상태에 의해서만 결정된다고 가정할 경우, 현재 상태에 대한 가치는 현재 상태와 액션에 대한 보상 값과 미래 보상의 누적합의 최대로 결정된다. 여기서 γ 는 감가율로, 미래에 더 좋은 상태를 가져오는 액션을 선택할 수 있도록 해준다.

본 연구에서 사용하는 기계 학습은 심층 강화 학습으로, 심층 강화 학습이란 강화 학습과 심층 신경망이 결합된 학습 방법이다.

2.1.4 DQN

DQN 은 Deep Q-Network 의 약자로, Google Deepmind 사에서 발표한 알고리즘 중의 하나이다[1]. CNN 을 이용하여 DQN 에 입력으로 사용되는 이미지를 전처리하고, target network, Q-network 두 개의 네트워크를 사용하여 학습을 진행한다. target network 는 매 epoch 마다 갱신되지 않고 Q-network 의 인자 값으로만 갱신된다.

강화 학습에서는 입력은 환경과 상호작용하며 차례로 들어오기 때문에 입력 간의 상관관계가 높고, 이러한 상관관계에 의해 잘못된 방향으로 학습이 진행될 수 있다. 따라서 입력 간의 상관관계를 없애기 위해 DQN 은 Experience Replay 를 이용하여 무작위로 이전 경험의 데이터를 뽑아서 학습을 진행한다.

Experience Replay 는 강화 학습을 진행하면서 매 순간의 상태, 액션, 보상 그리고 다음 상태를 데이터셋에 저장하고, 데이터셋에서 mini-batch 를 뽑아 학습시키는 방법을 의미한다[10]. Figure 6 에서 볼 수 있듯이 D 의 크기에 비해 정책이 오래된 경우 기존의

데이터로 학습을 진행하고 있어 정책의 갱신이 늦고, D 의 크기에 비해 정책이 오래되지 않았다면 새로운 데이터로 학습을 진행하고 있어 정책의 갱신이 빠른 것을 볼 수 있다. 따라서 데이터셋의 크기가 클수록 오래된 정책을 가지고 있기 때문에 Replay Capacity (D)와 Oldest Policy 를 조율하는 것이 필요하다.

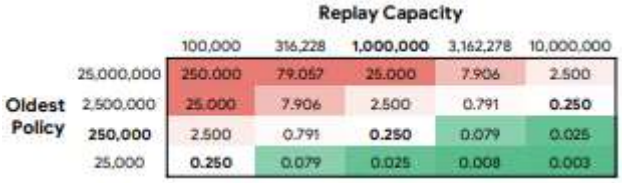


Figure 6 Replay Ratio[10]

2.2 경쟁 기술: Microsoft AirSim Autonomous Driving Cookbook[5]

AirSim 시뮬레이터를 이용한 자율 주행 자동차나 드론에 대한 다른 강화 학습 코드는 존재하지만, 본 논문의 경쟁 기술은 [5]로 특정한다. 이는 대부분의 강화 학습 코드들과 [5]의 강화 학습 방법이 DQN 으로 동일하고 입력과 출력 또한 유사하기 때문이다. 이외에도 [5]는 강화 학습을 위한 AirSim 시뮬레이터 Windows PowerShell 프로그램과 API 를 추가적으로 제공한다. 따라서 본 논문에서는 해당 경쟁 기술을 기존 모델로 사용하여 해당 기술과의 성능 평가를 진행한다.

[5]는 DQN 을 이용하여 학습한 자율 주행 자동차로, 액션에 대한 사전 학습 후 강화 학습을 진행하였다. 액션은 5 개의 이산 값으로 구성된 steering 값을 가지며 입력으로 한 장의 차량 전면 이미지를 가진다. 이에 대한 구체적인 내용은 다음 장에서 구체적으로 설명한다.

3 기존 모델

해당 연구는 [5]을 기반으로 한다. 또한 [5]에서 정의한 모델을 기존 모델로 명칭 한다. 기존 모델은 사전 학습을 진행한 후 심층 강화 학습의 Dqn 알고리즘을 기반으로 훈련하였다. 본 장에서는 기존 모델에 대한 신경망 구조, 보상 함수 등 모델을 구체적으로 설명한다.

3.1 사전 학습

기존 모델은 지도 학습을 통한 사전 훈련으로 얻은 가중치를 초기 가중치로 하여 심층 강화 학습을 진행한다.

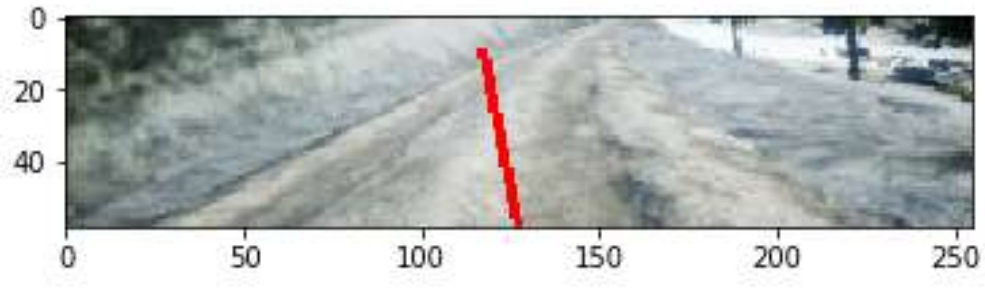


Figure 7 The Autonomous Driving Cookbook 지도 학습 데이터[5]

Figure 7 은 지도 학습에 사용한 데이터이다. 도로의 이미자와 자동차의 steering 을 이용하여 도로 모양에 따른 steering 값에 대한 사전 학습에 이용한다.

3.2 신경망 구조

신경망 구조는 입력층(input layer), 은닉층(hidden layer) 그리고 출력층(output layer)으로 구성되어 있다. Figure 8 은 기존 모델의 신경망 구조를 그림으로 나타낸 것이다.

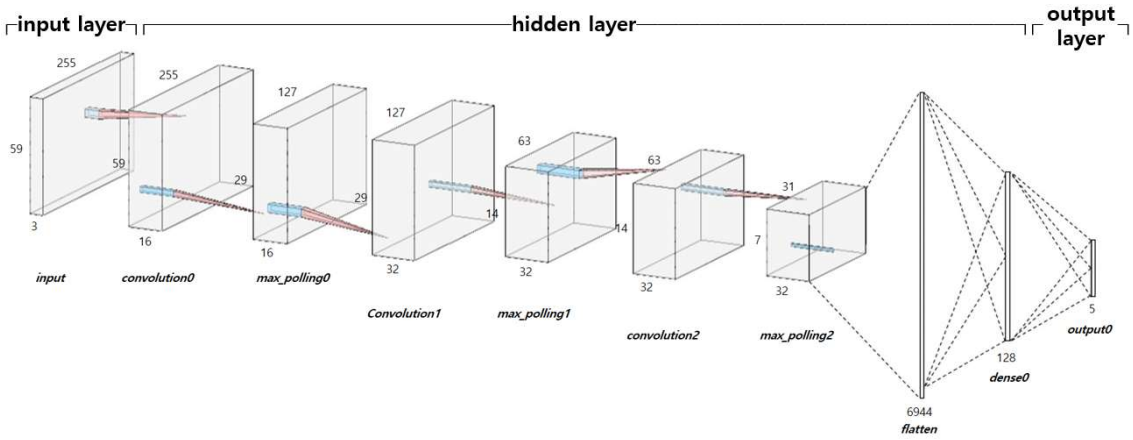


Figure 8 기존 모델 신경망 구조

3.2.1 입력층

먼저 기존 모델의 입력층의 깊이는 3 층이다. 이는 차량 전면 이미지를 RGB 이미지로 표현한 것으로, (59, 255, 3)의 모양을 가진다. Figure 9 는 실제 입력층에 입력되는 컬러 이미지를 나타낸 사진으로, 기존 모델은 해당 사진을 입력으로 가진다.

Figure 9 차량 전면 RGB 이미지

3.2.2 은닉층

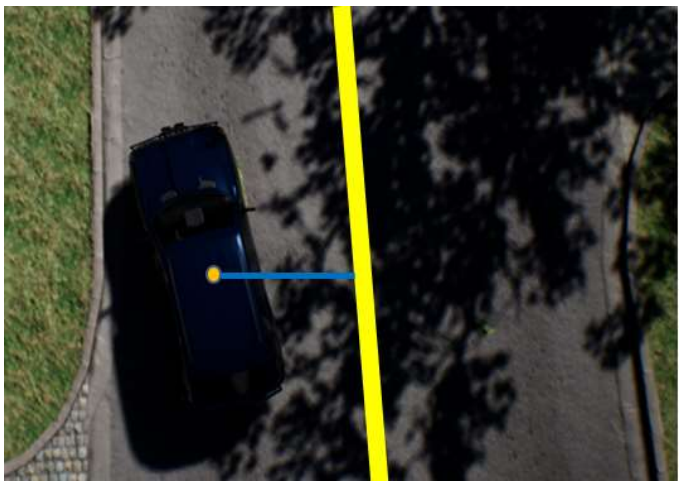
다음으로 기존 모델은 총 4 개의 은닉층을 가지고 있다. 이중 세 개는 CNN 층이며 나머지 한 개는 fully-connected Dense 층이다. Figure 8 에서 은닉층 구조를 확인할 수 있다.

3.2.3 출력층

심층 강화 학습에서 신경망의 출력은 액션을 의미한다. 기존 모델의 액션은 자동차의 steering 을 조작한다. steering 값은 이산 값으로 구성되어 있으며 [-1, -0.5, 0, 0.5, 1]의 값을 가진다. 따라서 기존 모델 신경망의 출력층은 5 개의 이산 값을 갖는 액션에 대한 Q 값을 출력으로 갖게 되므로 5 개의 뉴런으로 구성되어 있다.

3.3 보상 함수

기존 모델에서 보상은 거리에 대한 보상이다. Figure 10 을 보면 노란 선은 도로 중앙의 위치를 나타내며, 주황 점은 차량의 중심점을 나타낸다. 파란 선은 차량과 도로 중앙 사이의 거리를 나타내며, 이 거리를 이용하여 보상 값을 측정한다.



- d: 차량과 도로 중앙 사이의 거리
- γ: 거리 감가율
- R: 보상 값
- $R = e^{-d\gamma}$

Figure 10 기존 모델 - 보상 함수

$-d-\gamma:-R:-R=e^{-d\gamma}$ 기존 모델의 보상 함수는 차량과 도로 중앙 사이의 거리에 대한 함수로 이루어져 있다. d는 차량과 도로 중앙 사이의 거리로, 0~4 의 값을 가지고 있다. 총 보상은 e^{-d} 으로 얻어지며, γ는 거리 감가율로 d가 증가함에 따라 보상이 감소하는 비율을 의미한다. 따라서 거리 보상에서는 차량이 도로의 중심 점에서 멀어질수록 보상이 감소하고, 가까울수록 보상이 증가하게 된다.

3.4 한계점

본 논문에서는 기존 모델에 대한 세 개의 한계점에 관해 논한다.

첫 번째는 훈련 성능이다. [5]에서 제공되는 훈련이 완료된 모델은 Azure 를 이용한 다중 대행자 환경에서 훈련을 수행한 결과이다. 해당 코드를 local 환경에서 수행한 결과, 오랜 시간 학습시켰음에도 학습이 완료되지 않았다. 또한 특정 액션으로 출력이 수렴하는 결과도 나타났다.

두 번째는 자율 주행 시 기존 모델이 조절할 수 있는 특징이 steering 값 하나라는 것이다. 이는 단순히 핸들 각도만을 조절한다는 것인데, 자율 주행을 하기 위해서는 핸들 각도뿐 아니라 속도를 조절하는 등과 같은 추가적인 능력도 필요하다.

세 번째는 시각적 요소(전면 사진)에만 의존하여 다음 액션을 결정한다는 것이다. 이는 실제 자율 주행 시 주행에 영향을 미치는 다른 요소들을 고려하지 않는다는 한계를 가진다.

4 제안 모델

본 장에서는 기존 모델의 한계점을 극복하기 위해 제안된 모델들에 대해 설명한다. 본 논문에서는 총 두 개의 모델이 제안되었으며, Figure 11 은 기존 모델과 제안 모델들의 특징에 관한 표이다.

	기존 모델	제안 모델 1	제안 모델 2
입력층	차량 전면 사진	차량 전면 사진+핸들 사진	차량 전면 사진+핸들 사진+속도 사진
은닉층	3개의 CNN 층 + 1개의 Dense 층	3개의 CNN 층 + 3개의 Dense 층	3개의 CNN 층 + 3개의 Dense 층
출력층	절대 이산 값(steering) [-1, -0.5, 0, 0.5, 1]	절대 이산 값(steering) [-0.5, -0.25, 0, 0.25, 0.5]	상대 이산 값(steering) + [throttle, brake]
보상	도로의 중심과 차량 사이의 거리	도로의 중심과 차량 사이의 거리 + 도로의 방향과 차량의 방향 차이	도로의 중심과 차량 사이의 거리 + 도로의 방향과 차량의 방향 차이 + 속력의 보상
지도학습 기반	O	X	X

Figure 11 모델 비교표

4.1 제안 모델 1

제안 모델 1 은 기존 모델에서 제기되었던 테스트 성능의 향상을 위해 제안되었다. 또한, 입력으로 시각적 요소 이외에 다른 요소를 고려하는 방법을 제안한다. Figure 12 는 제안 모델 1 의 신경망 구조를 그림으로 나타낸 것이다.

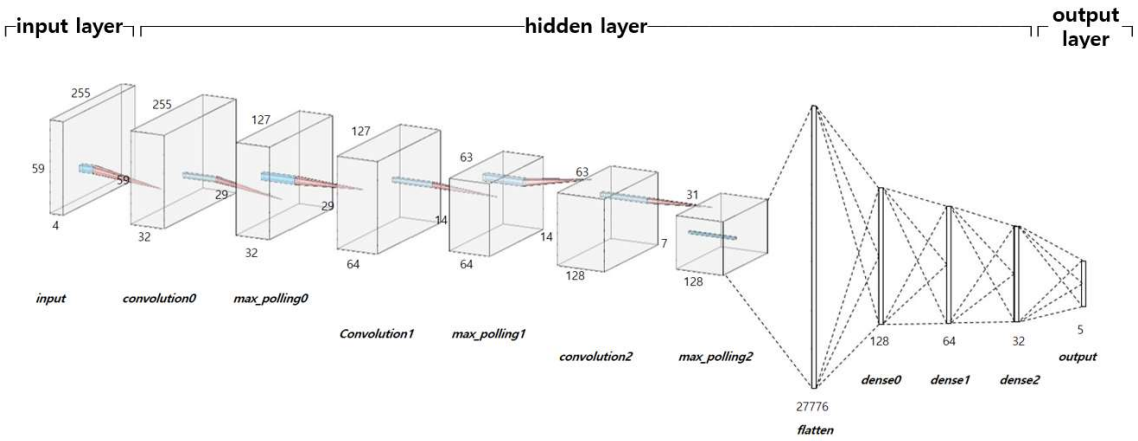


Figure 12 제안 모델 1 의 신경망

4.1.1 입력층

입력은 액션을 결정하기 위한 요소를 의미한다. 기존 모델은 차량 전면 이미지만 이용하여 액션을 결정하였기 때문에 입력이 차량 전면 RGB 이미지로 만 이루어져 있었다. 하지만 현실에서 사람은 운전할 때 시각적 정보에만 의존하여 차량을 운전하지 않는다. 차량의 핸들 각도, 차량의 속도 등 다른 요소를 고려하여 판단을 내린다. 이에 본 연구에서는 시각적 정보뿐 아니라 현재 차량의 핸들 각도를 고려할 수 있는 방법을 제안한다. 심층 강화 학습을 이용한 연구 중 가장 큰 성과를 거둔 [3]에서는 바둑돌을 놓을 차례, 과거의 상태 등 액션 결정에 영향을 주는 요소를 이미지화 시켜 입력으로 주었다. 즉 상태를 이미지화 시켜 기존의 입력에 쌓아주는 것이 효과를 거두었고, 본 연구에서는 이에 착안하여 핸들의 각도를 이미지화 시켜 입력층에 쌓는 방법을 제안한다. Figure 13 은 핸들의 각도를 이미지화 시켰을 때의 모습을 보여주며 기존 차량 전면 이미지에 해당 핸들 이미지를 쌓아 4 개의 층으로 입력층을 구성하였다.

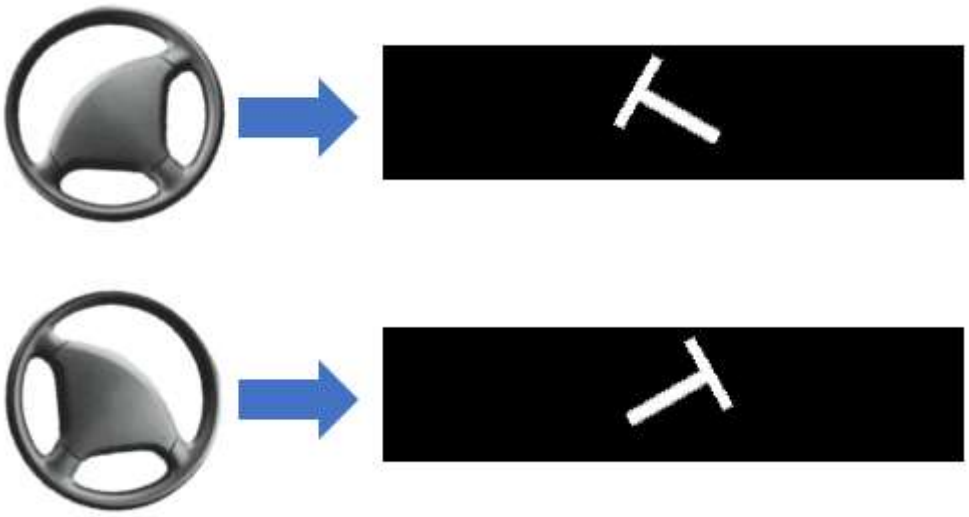


Figure 13 이미지화 시킨 핸들의 각도

4.1.2 은닉층

기존 모델을 local 에서 훈련시켰을 때 훈련이 거듭됨에 따라 액션이 하나의 값으로 수렴하는 문제가 있었다. 심층 신경망에서 은닉층의 깊이는 중요한 hyperparameter 이다. 신경망이 너무 얕아 학습이 잘 이루어지지 않을 수도 있고, 너무 깊어 학습의 속도 저하 및 과적합 등의 문제 또한 생길 수 있다. 본 연구에서 액션을 결정하기에 기존의 기존 모델에서 사용한 1 개의 dense 층은 액션 결정에 너무 얕아 특정 값으로 수렴하는 결과를 얻었다고 판단하여 은닉층의 깊이를 늘리기로 결정했다. 따라서 제안 모델 1 의 은닉층 구조는 3 개의 CNN 층과 3 개의 dense 층으로 구성된다.

4.1.3 출력층

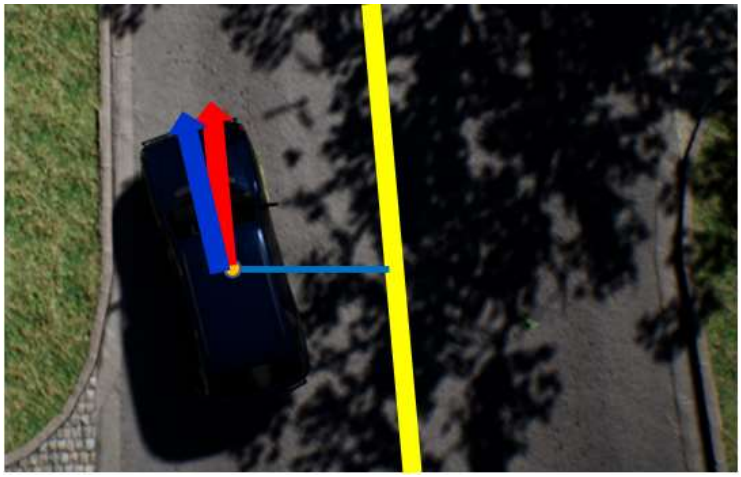
기존 모델의 출력은 이산 값으로 구성되어 있다. 이는 AirSim 시뮬레이터의 특징으로, steering 값을 특정 값으로 설정하면 해당 steering 값으로 차량의 steering 을 즉각적으로 변경할 수 있다. 기존 모델을 훈련하고 테스트해 본 결과 실제 주행할 때 핸들의 각도의 범위가 [-1, 1]의 범위를 주행에 사용하는 것의 필요성이 거의 없기 때문에 해당 범위를 [-0.5, 0.5]로 축소하였다.

4.1.4 보상 함수

강화 학습에서 보상은 훈련에 매우 큰 영향을 미치는 중요한 요소이다. 강화 학습에서 보상은 문제를 풀기 위한 목적에 해당하고 대행자는 해당 보상을 최대화하기 위해 훈련한다.

DQN 을 사용한 훈련은 모든 epoch 을 독립 시행으로 취급한다. 이는 각 epoch 당 경험들이 서로 연관성이 없는 경험임을 의미하고 각각의 경험에서 대행자는 해당 상태(입력)에서 특정 액션을 취했을 때 보상의 차이가 존재해야 한다는 것을 의미한다. 하지만 기존 모델과 같이 보상 함수를 거리에 대한 보상으로 설정하면 해당 시점에서의 거리 값은 어떤 액션을 취하던 같은 값을 갖기 때문에 해당 시점에서의 보상은 큰 차이를 가지지 않는다. 따라서 본 연구에서는 거리로만 보상을 주는 것이 아닌, 모델의 액션인 steering 값에 따라서 보상을 받을 수 있는 방법을 제안한다. 제안 모델 1 은 거리에 대한 보상 이외에 방향에 대한 보상을 정의하였는데, 이는 해당 시점에서 도로의 방향대로 이동하고자 할 때 보상을 더 주기 위함이다. 실제에서도 차량의 방향을 정해주는 차선, 도로의 모양 등으로 도로의 방향을 볼 수 있기 때문에 도로의 방향을 인식하기 위한 보상을 추가하였다.

기존 모델과 동일하게 Figure 14 의 노란 선은 도로 중앙, 노란 점은 차량의 위치, 파란 선은 도로 중앙과 차량 사이의 거리를 의미한다. Figure 14 의 빨간 화살표는 도로의 방향, 파란 화살표는 차량의 방향을 나타낸다.



$-o$: 도로 방향과 차량 방향의 차이

$-\gamma_o$: 방향 감가율

$-R_d$: 거리 보상 값

$-R_o$: 방향 보상 값

$-R$: 최종 보상 값

$-R_o = e^{o\gamma_o}$

$-\boldsymbol{R = \alpha R_d + (1-\alpha)R_o(1-\alpha)R_o}$

Figure 14 제안 모델 1 - 보상 함수

방향에 대한 보상 함수는 기존 모델의 보상 함수와 동일하지만 방향의 차이 o를 거리 d 대신 사용한다. o는 0~1 의 연속 값을 가지며 방향의 차이 0~180 도를 의미한다. o는 방향 감가율로, 도로와 차량의 방향 차이가 커질수록 보상의 차이를 얼마큼 빠르게 감소시킬지에 대한 정도를 의미한다. 총 보상은 거리 보상과 방향 보상의 합으로 각각의 보상의 얼마만큼의 가중치를 둘 것인지에 따라 값을 설정해 줄 수 있다. 본 논문에서는 거리 보상과 방향 보상의 비율을 똑같이 주었기 때문에 α를 0.5 로 설정하였다.

4.2 제안 모델 2

제안 모델 2 의 고안점은 속도 제어다. 기존 모델과 제안 모델 1 모두 steering 만을 제어하는 모델이다. 따라서 제안 모델 2 에서는 속도를 조절하는 방법을 제안한다. Figure 15 는 제안 모델 2 의신경망 구조를 그림으로 나타낸 것이다.

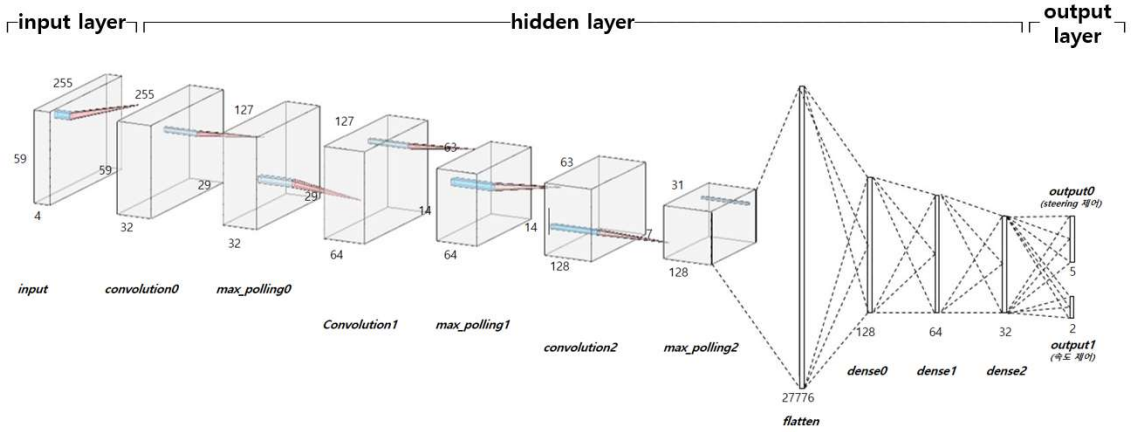


Figure 15 제안 모델 2 의 신경망

4.2.1 입력층

제안 모델 2 의 가장 중요한 특징은 앞서 설명했듯 속도 제어이다. 속도를 제어하기 위해서는 시각적 요소나 핸들의 각도 이외에도 다른 정보가 필요하다. 속도를 제어하는 데 있어 가장 중요한 정보는 현재의 속도이다. 속도를 제어하기 위해 현재 속도에 대한 정보를 가지고 있어야 속도를 조절할 수 있다. 따라서 입력으로 현재의 속력에 대한 정보를 함께 넣어줘야 한다. 제안 모델 1 에서와 마찬가지로 현재 속도에 대한 정보를 이미지화 시켜 입력층에 추가하는 방법을 제안한다. Figure 16 은 이미지화 시킨 속도의 스펙트럼이다. 최저 속도와 최고 속도 기준으로 속도가 상승함에 따라 색을 진하게 표현함으로써 속도를 이미지화 시켰다.

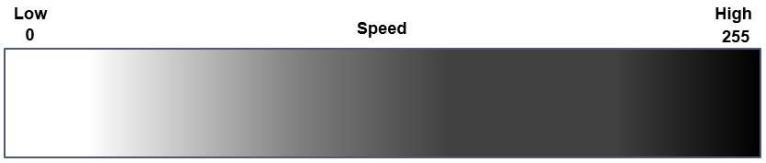


Figure 16 이미지화 시킨 속도의 스펙트럼

4.2.2 은닉층

제안 모델 2 의 은닉층 구조는 제안 모델 1 의 은닉층 구조와 동일한 구조를 사용한다.

4.2.3 출력층

제안 모델 2 는 steering 뿐만 아니라 속도 또한 조절한다. 따라서 신경망의 출력은 핸들 각도를 결정하는 부분과 속도를 조절하는 부분으로 나뉜다.

제안 모델 2 의 출력은 앞서 설명했듯이 핸들의 각도를 결정하는 부분과 속도를 조절하는 부분 두 가지로 나뉜다.

먼저 핸들의 각도를 결정하는 부분은 기존 모델, 제안 모델 1 과 구별되는 특징을 가진다. 앞선 두 모델은 절대적인 이산 값을 사용한다. 절대적인 이산 값의 의미는 해당 이산 값이 실제 핸들 각도로 설정된다는 것이다. 여기서 나오는 문제점은 차량이 취할 수 있는 steering 값이 액션의 개수, 즉 출력의 개수와 동일하다는 점이다. DQN 알고리즘은 이산 액션을 위한 알고리즘이기 때문에 가질 수 있는 액션 개수의 한계가 있다. 제안 모델 2 에서는 같은 이산 값이지만 앞선 방법에 비해 같은 액션 개수에서 더 많은 steering 값을 취할 수 있는 방법을 제안한다.

Figure 17 은 특정 핸들의 각도에서 액션을 취했을 때 변화되는 steering 값을 보여준다. 기존 모델, 제안 모델 1 과 동일하게 액션은 5 개로 동일하지만 이 값은 절대적인 값이 아니라 상대적인 값을 나타내기 때문에 현재 핸들의 각도에 따라 더 많은 값을 취할 수 있다.

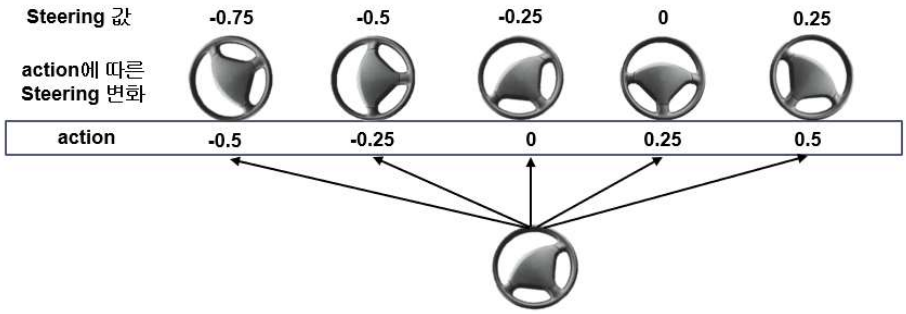
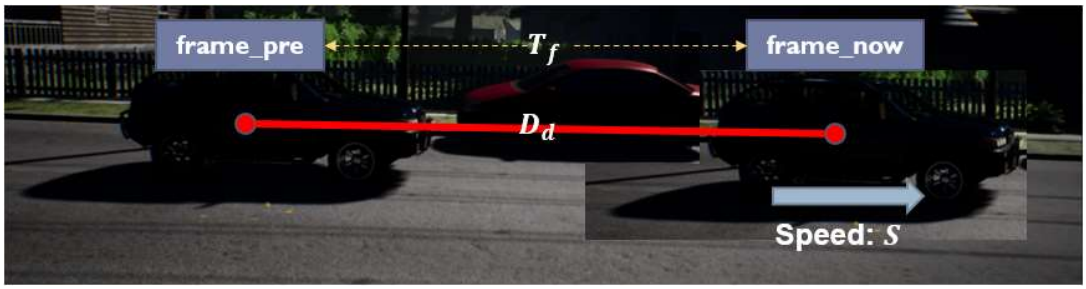


Figure 17 액션에 따른 실제 Steering 값

다음은 속도를 조절하는 부분이다. 해당 출력 부는 두 개의 값으로 구성된다. 하나는 throttle 을 의미하는 부분이고 다른 하나는 brake 를 의미한다. [1, 0]은 throttle 값을 1 로 함으로써 가속, [0, 1]은 brake 값 만을 1 로 함으로써 정지, [0, 0]은 물리적인 엔진을 없앤 상태로 물리적 감속을 의미한다. 이와 같이 3 개의 액션을 이용하여 차량의 속도를 조절할 수 있다.

4.2.4 보상 함수

출력층의 구조가 바뀜으로써 보상 함수도 변경이 필요하다. 그 이유는 특정 액션을 취함에 따라 보상 값이 바뀌어야 보상 값을 늘리기 위한 액션을 학습하게 되는데, 속도 조절에 대한 보상 값은 앞선 보상 함수로는 줄 수 없기 때문이다. 제안 모델 2 에서는 각각의 출력부에 서로 다른 보상 함수를 사용하였다. 핸들 각도를 조절하는 출력부에서는 제안 모델 1 과 동일한 보상 함수를 사용하였고, 속도를 조절하는 출력부에 속도에 대한 보상 함수를 새로 정의하였다.



$-\ S$: 현재 속도

$-\ T_f$: 단위 프레임당 시간

$-\ D_d$:단위 프레임당 이동거리

$-\ D_d = T_f * \max(0, S)$

$-\ D_{\max} = T_f * S_{max}$ $\quad -\ D_{\min} = T_f * S_{min}$

$if\ not\ stopped$:

$R_t = min - \max normalized\ D_d = \frac{D_d - D_{min}}{D_{max} - D_{min}}$

$else$:

$R_t = 0$

Figure 18 제안 모델 2 - 보상 함수

Figure 18 은 제안 모델 2 의 보상 함수의 알고리즘을 나타낸 그림과 수식으로, 그림에서 회색 화살표는 차량의 진행방향을 의미한다. S 는 현재 차량의 속도를 의미한다. 이때 보상 함수를 쉽게 구하기 위해 차량의 최대 속도를 10 m/s 로 제한하였다. T_f 는 단위 프레임당 시간을 의미한다. 해당 시간은 차량 주행에 있어 매번 작은 차이를 보였기에, 다량의 데이터를 모아 T_f 값의 평균값인 0.05 초를 이용하기로 한다. D_d 는 단위 프레임당 이동 거리이며, 거리는 속력과 시간의 곱임을 이용하여 구한다. 속도 조절의 리워드인 R_t는 차량의 상태에 따라 두 가지 경우로 나뉜다. 차량이 이동하고 있을 경우에는 단위 프레임당 시간을 min-max normalize 한 값을 R_t 로 사용한다. 반면 차량이 정차한 경우 R_t는 0 을 갖게 된다.

5 구현 및 성능 평가

5.1 구현

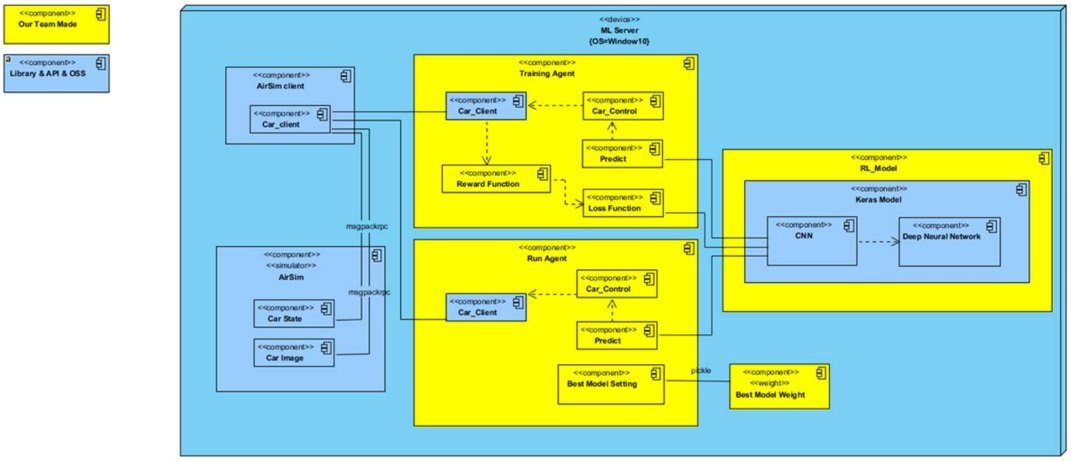


Figure 19 시스템 deployment diagram

Figure 19 는 본 논문의 시스템 deployment diagram 을 보여준다. 본 논문을 위한 노드는 훈련을 진행하는 서버 하나로 구성되어 있다. 훈련을 위해 필요한 구성 요소는 시뮬레이터(AirSim), 시뮬레이터와 통신하는 API (AirSim client), 대행자(Training Agent) 그리고 모델(RL_Model)로 구성되어 있다. 또한 모델을 동작하기 위한 대행자와 훈련이 완료된 가중치로 구성되어 있다. 본 논문에서 개발한 컴포넌트는 시뮬레이터, API 와 Keras 모델을 제외한 나머지이며, Figure 19 에서 노란 박스가 본 논문에서 개발한 컴포넌트에 속한다. 해당 시스템에 동작은 훈련 상황과 테스트 상황으로 나뉜다.

먼저 훈련 상황에 시스템 동작은 크게 AirSim, AirSim client, Training Agent 그리고 RL_Model 들로 동작한다. Training Agent 를 이용하여 훈련을 진행하며 Training Agent 는 Car_client 컴포넌트를 통해 AirSim client 와 통신하여 AirSim 시뮬레이터의 차량을 조작한다. RL_Model 을 통해 다음 액션을 예측하게 되고 해당 액션 이후 상태를 통해 받은 보상을 토대로 다시 RL_Model 을 학습시킨다.

다음은 테스트 상황이다. 테스트 상황에는 Training Agent 대신 Run Agent 를 사용하며 훈련이 완료된 가중치인 Best Model Weight 를 이용하여 RL_Model 의 가중치를 업데이트 한 뒤 모델을 동작한다. 테스트 상황은 예측과 조작으로 진행되며 RL_Model 을 이용하여 예측한 뒤 차량을 조작하여 주행하게 된다.

5.2 성능 평가

본 장에서는 본 논문에서 제안한 모델의 성능 평가를 진행한다. 성능 평가는 기존 모델과 제안 모델 1 의 훈련과 테스트 성능에 대해 진행하며, 제안 모델 2 는 훈련 시간이 지정된 epoch 종료 조건인 30 초 주행을 만족하지 못하였기에 성능 평가를 실시하지 않는다. 성능 평가를 진행한 두 모델은 모두 local 환경에서 훈련하였으며, 30 초 주행을 종료 조건으로 훈련하였다.

훈련 성능을 평가하기 위해서, 한 번의 epoch 에서 가장 긴 주행 시간을 best driving time 으로 저장하였다. 그리고 이전의 best driving time 보다 더 오래 주행했을 경우, 해당 시점의 시간과 주행 시간을 저장하고 best driving time 을 갱신하였다. 주행 시간이 30 초를 넘었을 경우, best driving time 을 30 초로 저장한 뒤 훈련을 종료하였다.

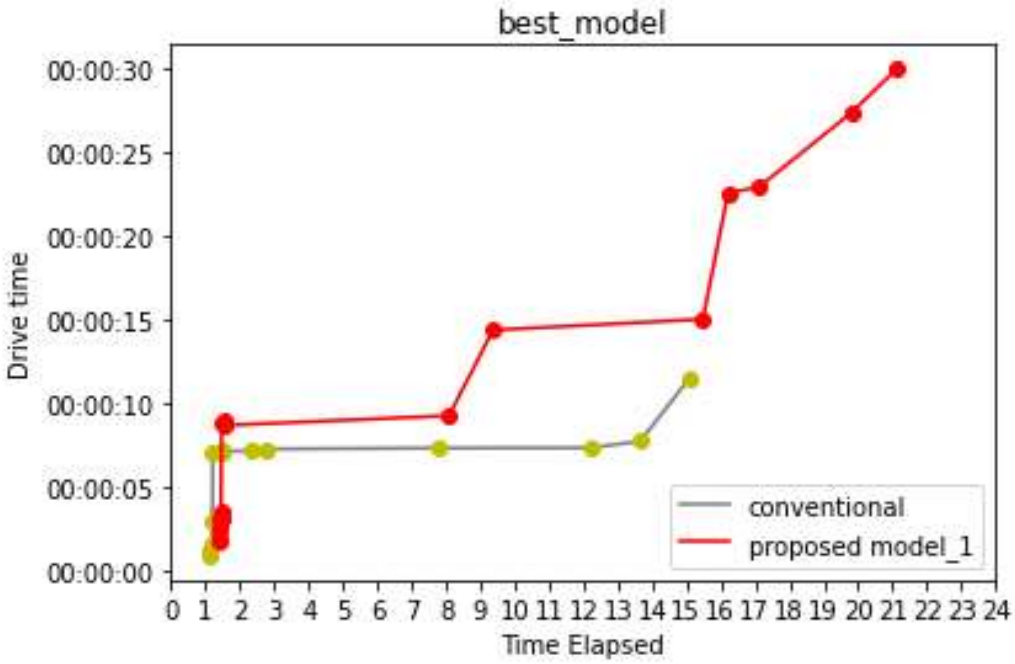


Figure 20 훈련 성능 평가

Figure 20 은 기존 모델과 제안 모델 1 을 훈련하면서 한 번의 훈련 동안 저장된 best driving time 들을 그래프로 나타낸 것이다. 그래프의 각 점들은 best driving time 이 갱신된 시점을 의미한다. 기존 모델의 훈련은 15 시간이 경과했을 때 best driving time 이 약 11 초를 기록했고, 24 시간이 경과하여도 best driving time 이 11 초를 넘지 못하였다. 그러나 제안 모델 1 의 훈련은 9 시간이 경과하였을 때 이미 기존 모델의 best driving time 인 11 초를 넘어섰고

21 시간이 경과하였을 때는 기존에 제안했던 종료 조건인 30 초 주행을 완료하였기에 훈련을 종료하였다.

테스트 성능을 평가하기 위해 동일한 시간 동안 훈련을 실시한 기존 모델과 제안 모델 1 을 사용하였다. 테스트 시 출발지는 AirSim 환경의 neighborhood map 내에서 무작위로 설정한다. 이후 모델로 동작하는 AirSim 환경의 차량이 지형지물에 충돌할 때 까지의 시간을 driving time 으로 한다. Figure 21 은 각각의 모델을 10 번 테스트한 후 각각의 driving time 으로 box plot 을 그린 것이다. 왼쪽의 파란색 box plot 은 제안 모델 1 의 테스트 결과를 사용하였고, 오른쪽의 주황색 box plot 은 기존 모델의 테스트 결과를 사용하였다. 제안 모델 1 의 테스트 최대 주행 시간은 46 초이며 평균 주행 시간은 28 초이다. 테스트 시 기존 모델의 최대 주행 시간은 15 초이며 평균 주행 시간은 9 초이다.

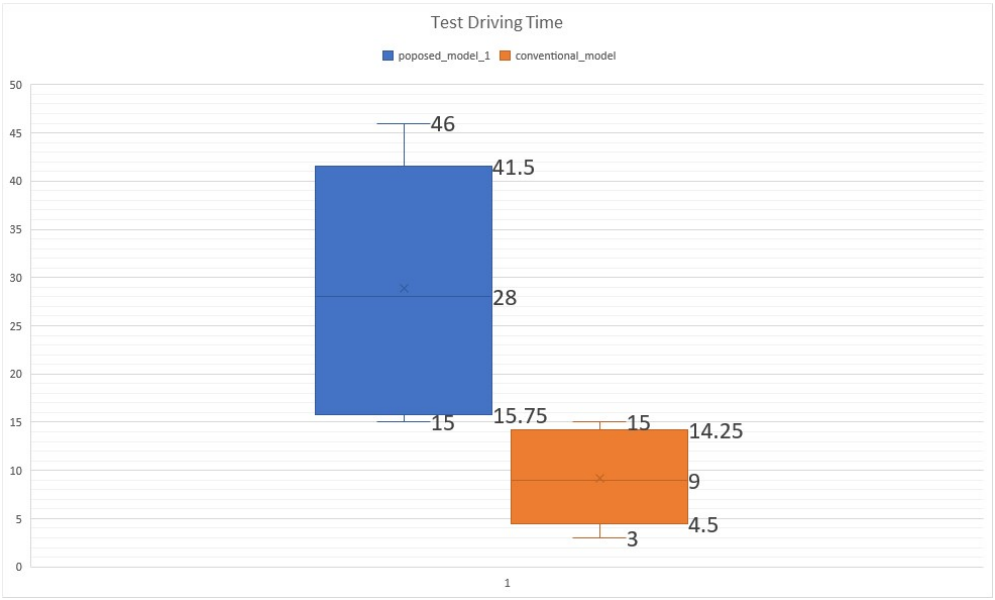


Figure 21 테스트 성능 평가

본 장에서는 훈련과 테스트 성능을 이용하여 성능 평가를 진행했다. 24 시간 동안 훈련한 best driving time 을 비교한 결과 제안 모델 1 의 훈련 성능이 기존 모델에 비해 273% 상승하였다. 또한 기존 모델의 best driving time 을 기준으로 제안 모델 1 이 학습 시간을 37.5% 단축하였으며, 기존 모델은 라벨링 된 이미지들을 이용하여 사전 훈련시킨 가중치를 이용한 모델을 사용하기에 사전 훈련에 소요된 훈련 시간까지 고려한다면 훈련 시간의 차이는 더욱 벌어진다. Figure 21 에서는 같은 시간 동안 훈련시킨 두 개의 모델을 이용하여 테스트 성능 평가를 한 결과 제안 모델 1 이 기존 모델에 비해 310% 향상된 성능을 가진다는 것을 확인할 수 있다.

6 한계점

1. 제안 모델 1

제안 모델 1 의 액션은 이산 값으로 구성된다. 따라서 주행 시 steering 값이 가질 수 있는 값의 범위가 한정된다. 이는 실제 주행 시 차량의 움직임이 자연스럽지 않는다는 한계가 존재한다. 또 다른 한계점은 액션이 steering 조절로 한정되어 있다는 점이다. 이는 속도 조절을 못한다는 점에서 큰 한계를 가진다. 제안 모델 1 은 속도를 조절하지 않고 특정 속도 범위 내에 유지한다. 따라서 특정 상황(좌회전이나 우회전 상황, 장애물 감지 등)에 속도를 조절하지 못한다.

2. 제안 모델 2

제안 모델 2 는 기존 모델과 제안 모델 1 과 다르게 액션 값을 절대적인 값이 아닌 상대적인 값으로 구성함으로써 취할 수 있는 steering 값의 범위를 늘렸다. 하지만 steering 값은 이산 값으로 구성되어 있다는 한계를 가진다. 다음으로는 성능의 한계이다. 모델을 설계한 뒤 훈련하는 과정에서 훈련이 잘 이루어지지 않았는데, 훈련할 때 자동차가 정지해 있는 문제가 자주 발생한다. 이는 대행자가 정지해 있는 것이 보상을 최대로 받는다고 학습했기 때문이다. 이러한 문제는 보상 함수가 잘못 정의되었음을 의미한다. 제안 모델 2 에서 출력부는 두 부분으로 구성되었고, 각 부분에 다른 보상 함수가 사용되었다. 하지만 두 출력부는 서로 연관성이 있는 출력이고, 이러한 복잡한 출력부 구조에 따른 보상 함수를 정의하기 어렵다는 한계를 가진다.

3. 시뮬레이터

- 그림자: 그림자가 도로 모양에 영향을 미쳐서 학습에 어려움을 겪었다.
- 학습 환경의 다양성 부족: AirSim 시뮬레이터의 환경 중 우리가 직접 제어할 수 있는 환경이 부족하여 다양한 환경에서 학습을 실시하지 못하였다.
- 상황 설정의 불가능: AirSim 시뮬레이터의 환경에서 장애물이나 날씨를 임의로 조절하는 것이 불가능했다. 그렇기에 다양한 환경을 제공하는데 어려움이 있었다.

4. 강화 학습을 이용한 자율 주행

자율 주행은 단순한 판단이 아닌 복합적인 판단을 요구한다. 강화 학습을 이용하여 특정 행동들을 훈련시키는 것은 가능하지만 복합적인 판단을 훈련시키는 것에 한계가 존재한다. 또한 본 논문에서 사용한 DQN 은 이산 액션을 위한 학습 방법이라는 점에서 한계가 존재하며, 시뮬레이션 환경이 필수적인 강화 학습에서 자율 주행의 예기치 못한 상황을 시뮬레이션 환경에서 얻기 힘들다는 한계도 있다.

7 결론

본 논문에서는 AirSim 시뮬레이터 환경에서 자율 주행을 위한 심층 강화 학습을 연구하였다. 기존 모델을 기반으로 더 좋은 성능을 얻기 위해 액션, 신경망, 보상 함수를 향상시킨 제안 모델 1 과 속도 조절 기능을 추가한 제안 모델 2 를 제시하였다. 제안 모델 1 과 기존 모델의 훈련 성능과 테스트 성능을 비교해 본 결과, 제안 모델 1 의 성능이 압도적으로 좋은 것을 확인할 수 있었다.

그러나 제안 모델 2 의 경우 원하는 수준의 성능을 얻지 못하였다. 이는 6 장의 한계점에서 설명한 요인 때문이라고 생각한다. 따라서 향후 계획으로 제안 모델 2 의 이러한 요인을 제거하기 위한 연구를 계속 할 것이다.

8 참고 문헌

[1] V. Mnih, K. Kavukcuoglu, D. Silver, et al. “Human-level control through deep reinforcement learning,” Nature 518, pp. 529–533, 2015.

[2] 정보통신 산업진흥원, “NIPA 이슈리포트 2017 제 10 호 - 국내외 동향을 통해 살펴본 국내 자율주행차 산업의 개선점,” https://www.itfind.or.kr/publication/regular/periodical/read.do?selectedId=02-001-170816-000001, 2017.

[3] D. Silver, J. Schrittwieser, K. Simonyan, et al. “Mastering the game of Go without human knowledge,” Nature 550, pp. 354–359, 2017.

[4] J. Schrittwieser, I. Antonoglou, T. Hubert, et al. “Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model,” Preprint at https://arxiv.org/abs/1911.08265, 2019.

[5] Microsoft, “The Autonomous Driving Cookbook,” https://github.com/microsoft/AutonomousDrivingCookbook

[6] Computer Vision Center, “Carla,” https://carla.org/

[7] Parallel Domain, “Deep Drive,” https://deepdrive.io/index.html

[8] Y. LeCun, Y. Bengio and G. Hinton, “Deep learning,” Nature 521, pp. 436–444, 2015.

[9] S. Kim, “강화학습을 위한 심층신경망 구조 분석,” 정보산업공학논문집 제 36 집, 2020.

[10] W. Fedus, P. Ramachandran, R. Agarwal, et al. “Revisiting Fundamentals of Experience Replay,” Preprint at https://arxiv.org/abs/2007.06700, 2020.