Pusan National University

Computer Science and Engineering

Technical Report 2024-10

강화학습기반 교차로 동적 신호제어시스템

200628104 권오성

202155652 이준표

201924578 정하립

지도교수　박진선

목 차

1. 서론

1.1. 연구 배경

평상시 운전을 하다 보면 교차로에서 신호를 기다리는 일이 많다. 수많은 차량과 보행자는 도로 위에 설치된 신호등이 제어한다. 신호 체계에 흥미를 느껴 조사해본 결과, 교차로의 각 방면별로 주어지는 신호 순서와 신호 길이가 교통량과 관계없이 고정적이라는 것을 발견했다. 즉, 해당 방면에 대기하는 차량의 수와 관계없이 신호등은 일정한 주기로 각 방면에 일정한 시간만큼의 신호를 준다. 평소에 일정하고 예측이 뚜렷이 가능한 교통량이 보장된다면 상관없겠지만 일시적으로 특정 방면으로의 차량수가 늘어나거나, 혹은 줄어드는 상황에는 합리적으로 처리하기 위한 대안의 필요성을 느꼈다.

신호 제어와 관련된 자료를 조사하니, 국내에서 일반적으로 사용하는 방식은 **"시간제어식 신호"** 이다. **시간제어식 신호란,** 특정 상황별로 주기를 미리 지정해두고, 해당 상황이 발생하면 인위적인 조작을 통해 적용하는 방식이다. 이 방식은 기본적으로 정해진 고정된 여러 옵션에서 단순히 선택을 하는 방식이기 때문에, 정밀한 최적화가 힘들고, 인위적인 제어가 필요하기 때문에 실시간으로 대응하기에는 한계가 있다. 더 좋은 방식을 모색하던 중, 요즘 대부분의 분야에서 접할 수 있는 인공지능을 신호 제어에 적용시켜, 최적화된 신호체계를 만들면 효율적일 것이라는 발상을 했다. 교차로 데이터를 학습한 인공지능을 통해 실시간으로 교차로의 상황에 맞는 신호를 줄 수 있겠다고 생각했다. 이를 계기로, 강화학습기반 교차로 동적 신호제어시스템을 이번 프로젝트의 주제로 정하게 되었다.

1.2. 기존 문제점

1.2.1. 교통 상황을 반영하지 못하는 불합리한 차량대기 시간

교차로에서 차량 대기 시간이 필요 이상으로 길거나, 반대편 도로에 통행하는 차량 수가 적은 경우에도 불구하고 빨간 신호가 지속되는 경우가 종종 발생한다. 또 야간이나 새벽에는 통행 차량이 거의 없음에도 불구하고 신호등 앞에서 오래 기다려야 하는 경우가 있다. 이런 비효율적인 교통 신호 제어로 인해 경제적, 시간적인 손실이 발생하고, 운전자는 지나치게 긴 대기 시간으로 인한 피로감을 느낄 수 있다.

1.2.2. 고정된 순서로만 진행되는 신호 체계

일반적으로 교차로에서는 신호순서가 고정 되어있다. 예를 들어, "직진 후 좌회전" 신호를 사용하는 교차로의 경우 남, 북 도로 직진 → 남, 북 도로 좌회전 → 동, 서 도로 직진 → 동, 서 도로 좌회전의 순서로 신호순서가 고정되어 반복된다. 교차로 각 방면의 대기중인 교통량을 고려하지 않고, 고정된 순서대로만 신호가 바뀌면, 특정 방면에만 차량이 많은 경우, 고정된 신호 순서로 인해 필요 이상으로 기다려야 하는 문제가 발생합니다. 신호체계를 실시간으로 동적 제어할 경우, 유연한 신호 순서로 더 효율적인 교통 체계를 만들 수 있다.

1.2.3. 현행 신호 체계의 효율성과 유연성 부족

현재 국내에서 사용되는 신호 체계는 특정 상황별로 지정된 주기를 지정해 두고 해당 상황이 발생하면, 적용하는 **시간제어식** 신호를 주로 사용한다. 하지만 이 방식은 교통량의 변동에 따라 해당 구간의 신호에 대한 인위적인 조작이 필요하므로 비효율적이며, 빠른 대응이 어렵고 여러 구간에 걸친 미세 조정이 어렵다. 또한 횡단보행자를 고려하지 않는 문제점도 있다. 따라서, 인구가 밀집되고, 차량 수가 폭발적으로 늘어난 현대 도시에서는 불확실성이 증가하여, 보다 동적이고 유연한 신호 체계가 필요하고, 이를 위해 인공지능을 활용할 수 있다.

1.3. 연구 목표

1.3.1. 불필요한 대기시간을 최소화하는 합리적 신호체계 구현

교차로에 카메라를 설치하여 각 방면 도로를 실시간으로 인식하고, 특정 방면에 차량들이 (다수 혹은 오래) 대기하고 있으나 다른 방면에 오가는 차량이 적을 경우, 해당 방면에 녹색 신호를 주거나 신호 길이를 늘리는 등의 동적인 신호 체계를 설계한다. 즉, 실시간 감시 시스템을 구성하고, 교통량을 평가하여 동적으로 교차로의 신호를 제어할 수 있는 시스템을 설계한다.

1.3.2. 동적인 신호순서를 적용하여 효율성 증대

고정된 신호 순서를 사용하는 기존 체계를 탈피한다. 다른 신호들의 순서가 다 끝나기를 기다릴 필요 없이, 교통량이 많이 발생하는 방면의 신호로 즉시 변경할 수 있는 동적 신호 제어 시스템을 구현한다.

1.3.3. 여러 교차로의 교통상황을 고려한 실시간 최적화

보다 효율적이고 유연한 교통 신호 체계를 구축하기 위해서, 한 교차로의 자동차 통행은 물론, 주변 인접 교차로들의 통행량도 함께 고려해야 한다. 이를 위해 각 교차로에 설치된 카메라를 활용하여 실시간으로 위치별 차량 수와 대기 큐 길이 등의 데이터를 수집한다. 이렇게 수집한 데이터를 클라우드 환경에서 실시간으로 강화학습 모델로 추론한 후, 각 교차로 구간별 최적의 신호 상태를 도출하여 도시 전체의 신호 시스템에 적용하면, **거시적인 교통 로드 밸런싱**이 가능하다.

2. 연구 배경

2.1. 관련 연구

2.1.1. 다중 교차로환경에서 강화학습 기반 분산 교통신호 제어[1]

위 연구에서는 단일 교차로 환경과 다중 교차로 환경에서 교통 신호를 제어하는 두 가지 모델을 설계한다. 하나는 녹색 신호 순서 할당 모델로, 이는 이동 신호를 결정하여 차량의 흐름을 제어하는 모델이다. 다른 하나는 녹색 신호 시간 할당 모델로, 이는 녹색 신호의 지속 시간을 결정한다. 이 모델들은 강화학습을 사용하여 교차로의 상황을 인식하고, 다양한 매개변수들의 활용성과 유용성을 학습한다. 이를 통해 차량의 흐름을 제어하고, 교차로에서의 처리량을 최대화하는 것을 목표로 한다.

모델은 MDP (Markov decision process)를 사용하여 상태, 행동, 그리고 보상을 정의한다. 상태는 교차로 방향의 개수로 정의되며, 행동은 동시에 녹색 신호를 할당할 수 있는 방향의 집합으로 정의된다. 보상은 큐 길이의 표준편차와 처리량으로 구성된다.

큐 길이의 표준편차는 교차로에 있는 차선의 길이가 얼마나 흩어져 있는지를 나타내며, 처리량은 교차로를 지나간 차량의 개수를 나타낸다. 이 두 가지 요소를 최적화함으로써, 교차로에서의 차량 흐름을 효율적으로 제어하고, 처리량을 최대화하는 것이 가능해진다.

2.1.2. 강화학습 기반 교통신호 제어 시뮬레이션[2]

위 연구에서는 상태, 행동, 보상을 다음과 같이 정의한다. 상태는 각 차선에 정지한 차량의 수, 현재 신호의 등화 여부, 그리고 현재 등화된 신호의 경과 시간으로 정의된다. 이러한 정보는 에이전트가 어떤 행동을 취할지 결정하는 데 사용된다.

행동은 현재 신호를 유지할지, 아니면 다음 신호로 변경할지를 결정하는 것이다. 이 모델은 신호의 순서를 유지하고, 각 신호는 한 번의 신호 주기 동안 최소 한 번은 등화되어야

[1] Hyunjin Joo, & Yujin Lim (2020). Distributed Traffic Signal Control at Multiple Intersections Based on Reinforcement Learning.

[2] Jawoon Gu, Minhyuck Lee, & Chulmin Jun (2020-11-20). Traffic signal control simulation based on reinforcement learning.

한다는 제약 조건을 가지고 있다.

보상은 행동에 따라 변경된 교통 흐름을 모델에 제공한다. 보상은 정지 차량 대비 통과 차량의 비율로 정의되며, 이를 통해 정지 차량이 줄고, 통과 차량이 많아질수록 에이전트가 받는 보상이 커지게 된다.

2.1.3. 관련 연구와의 공통점 및 차이점

공통점

- 두 연구와 이 프로젝트와 MDP 구조를 사용하여 강화학습으로 최적의 정책을 학습하도록 한다.
- 모두 교차로나 도로 상황을 실시간으로 반영하여 동적 제어를 목표로 한다.

차이점

- 두 연구에서는 큐 길이의 표준편차 혹은 정지 차량 대비 통과 차량의 비율을 보상으로 설정하지만, 이 프로젝트에서는 객체 탐지 및 추적을 이용해 보상에 반영한다. 실시간 처리 측면에서 더 유리할 것으로 판단된다.
- 이 프로젝트에서는 Q-Learning 을 적용하여 최적의 정책을 찾으며, 실시간 데이터에 더 집중한다.

2.2. 배경 지식

2.2.1. 객체 탐지 (Object Detection) 모델

- **YOLOv8 (You Only Look Once)**

  YOLO(You Only Look Once)는 합성곱 신경망을 활용하는 일련의 고급 실시간 객체 감지 시스템으로, 2015 년 조셉 레드몬 등이 처음 소개한 기술이다. 탐지를 위해 여러 번의 forward pass 가 필요한 R-CNN 과 같은 이전의 Region Proposal Network 과 달리, YOLO 는 네트워크를 한 번만 통과하면 예측을 수행할 수 있다. 입력 이미지를 그리드 셀로 나누고 각 셀이 Bounding Box 및 관련 클래스 확률을 예측하는 방식으로 작동한다. 이 방식은 전체 이미지를 한 번에 효율적으로 처리할 수 있어 자율 주행이나 감시 시스템과 같이 빠른 대응이 필요한 어플리케이션에 특히 적합하다.

- **SAHI(Slicing Aided Hyper Inference)**

  SAHI(Slicing Aided Hyper Inference)는 대규모 및 고해상도 이미지에서 객체 탐지 알고리즘을 최적화하기 위한 라이브러리이다. SAHI 는 이미지를 처리 가능한 조각으로 나누고 각 조각에서 객체 탐지를 수행한 후 결과를 재조합하여 높은 정확도를 유지하며 메모리 사용을 최적화한다. **슬라이스 추론(Sliced Inference)** 기법을 통해 작은 조각에서 객체 탐지를 하여 계산 부담을 줄이고 탐지 품질을 보존한다. 의료 이미지나 위성 이미지와 같이 대규모 데이터 세트를 다룰 때와 이미지의 가장자리에서 객체를 감지하기 힘든 상황에서 유리하다.

- **ByteTrack 객체 추적(Object Tracking) 알고리즘**

  ByteTrack 은 다중 객체 추적(MOT)에서 탐지 성능을 극대화하기 위해 설계된 최신 알고리즘으로, 기존 방법들이 인식 점수가 임계값 이상인 탐지 박스에만 의존하여 임계값 미만인 탐지 박스는 제거하던 문제를 해결하기 위해 설계되었다. ByteTrack 은 모든 탐지 박스를 연관시켜 낮은 점수의 탐지 박스라도 유사성을 활용하여 객체를 복구하고 배경 탐지를 필터링한다.

2.2.2. 강화학습 알고리즘

DQN(Deep Q-Network)은 전통적인 Q-러닝 알고리즘에 심층 신경망을 접목하여 상태 공간이 크거나 연속적인 경우에도 효과적으로 학습할 수 있는 방법론이다. **Q-러닝**은 에이전트가 주어진 상태에서 최적의 행동을 선택하도록 학습하는 강화학습 기법으로, Q-값(Q-value)을 통해 행동의 가치를 평가한다. Q-러닝의 목표는 상태-행동 쌍에 대한 Q-함수 $Q(s,a)$를 학습하는 것이며, 이때 최적의 Q-값은 **벨만 방정식**을 통해 정의된다.

$$Q(s,a) = r + \gamma \max_{a'} Q(s',a')$$

[수식 1] 벨만 방정식

여기서 s와 a는 각각 현재 상태와 취한 행동, s'와 a'은 다음 상태와 다음에 취할 행동을 의미한다. r은 현재 상태에서 얻은 보상, γ는 미래 보상을 얼마나 고려할지 결정하는 감가율 (Discount factor)이다. 이 방정식은 현재 상태에서 행동을 취한 후의 즉각적인 보상 r과, 미래 상태에서 얻을 최대 보상 $\max_{a'} Q(s',a')$를 더하여 Q- 값을 계산한다. 그러나 상태-행동 공간이 매우 큰 경우, Q-Table 을 직접적으로 저장하기 어렵고 학습이 비효율적이다. 이를 해결하기 위해 DQN 은 심층 신경망을 사용하여 상태-행동 쌍을 입력 받아 Q-값을 근사한다. 신경망이 Q-값을 근사함으로써 Q-Table 대신 학습된 네트워크를 통해 최적 정책을 찾을 수 있다.

DQN 의 핵심 알고리즘은 에이전트가 환경과 상호작용하며 얻은 경험을 바탕으로 Q-값을 추정하는 것이다. 에이전트는 주어진 상태에서 행동을 선택하고, 이에 따른 보상을 받고 다음 상태로 전이한다. 이 과정에서 발생한 상태, 행동, 보상, 다음 상태 정보를 이용하여 네트워크의 가중치를 업데이트한다. 특히 DQN 에서는 위 벨만 방정식을 기반으로 손실 함수를 정의하고, 이 손실을 최소화하기 위해 역전파 알고리즘을 통해 신경망을 학습시킨다. 학습 과정에서 에이전트는 탐색(exploration)과 활용(exploitation)을 적절히 조절하여 환경에 대한 지식을 축적하고, 최종적으로는 최적의 교차로 신호 제어 정책을 학습하게 된다.

DQN 의 학습 성능을 크게 향상시키는 두 가지 중요한 기술 중 하나는 **리플레이 버퍼(Replay Buffer)**이다. 리플레이 버퍼는 에이전트가 환경과 상호작용하는 동안 얻은 경험을 저장하는 메모리로, 이 경험들을 무작위로 추출하여 학습에 사용한다. 이를 통해

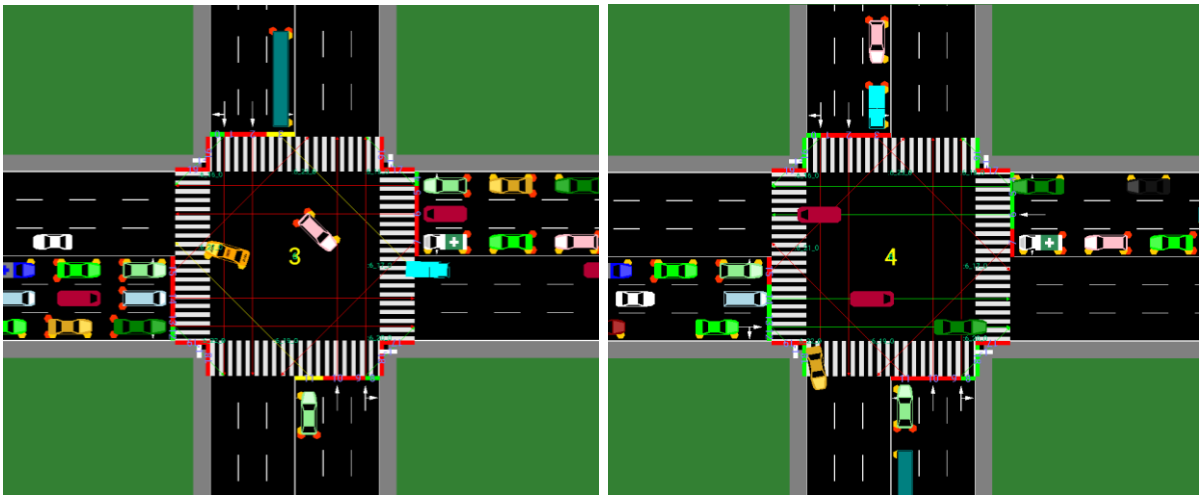
데이터의 상관성을 줄이고, 네트워크가 특정 순서의 데이터에 과적합되는 것을 방지한다. 만약 에이전트가 경험한 데이터가 시간 순서대로 네트워크에 입력된다면, 네트워크는 시간적으로 연속된 경험에 과도하게 적응할 수 있는데, 리플레이 버퍼는 이를 완화해준다. 또한, 버퍼 크기가 충분히 클 경우 다양한 경험을 쌓을 수 있어 일반화 성능이 향상된다. DQN 에서는 경험을 무작위로 샘플링하여 배치 학습을 수행함으로써, 데이터의 독립성 가정을 유지하면서 안정적으로 학습할 수 있다.

타겟 네트워크(Target Network)는 DQN 의 또 다른 중요한 기법으로, 학습의 안정성을 높이는 데 기여한다. 일반적으로 Q-러닝에서 Q-값 업데이트는 벨만 방정식을 기반으로 이루어지는데, 이때 Q-값의 업데이트는 Q-값 자체에 의존한다. 따라서 네트워크가 빠르게 업데이트되면, 목표값(Target)도 빠르게 변동하여 학습이 불안정해질 수 있다. 이를 방지하기 위해 DQN 은 두 개의 네트워크를 사용한다. 하나는 Q-네트워크로, 실제 학습을 통해 가중치를 업데이트하는 네트워크이며, 다른 하나는 일정한 간격으로만 업데이트되는 타겟 네트워크이다. 타겟 네트워크는 일정한 주기로 Q-네트워크의 가중치를 복사하여 고정된 상태로 학습에 사용되므로, 목표값의 급격한 변동을 억제하고 학습의 안정성을 보장한다. 이처럼 두 네트워크를 활용한 구조는 학습의 수렴성을 높이고, 최적의 Q-값을 효과적으로 찾을 수 있도록 도와준다.

2.2.3. 교통 시뮬레이터

- **SUMO**

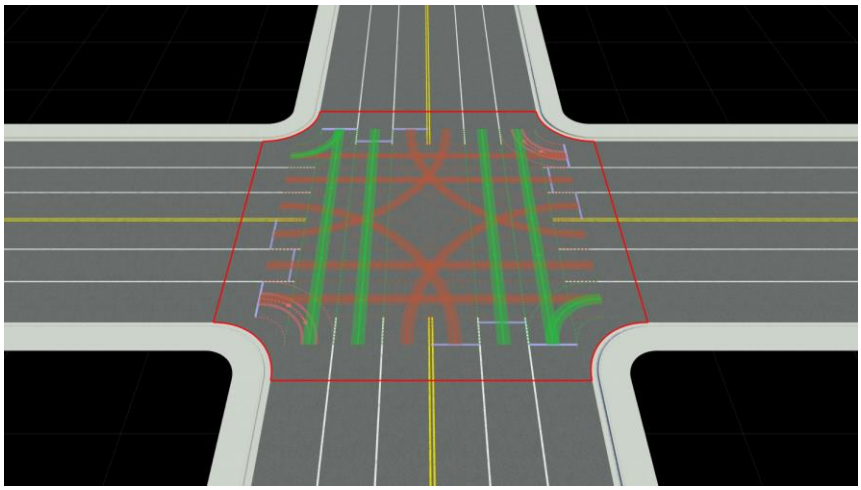
  SUMO 는 도시 교통의 시뮬레이션을 위한 오픈 소스 소프트웨어다. 대규모 네트워크를 처리하기 위해 설계된 미시적이고 연속적인 다중 모드 교통 시뮬레이션 패키지로, 도로 차량, 대중교통, 보행자 등의 교통 시스템을 모델링할 수 있다. 또한 경로 찾기, 시각화, 네트워크 가져오기, 배출량 계산 등의 다양한 도구를 지원한다. 사용자 정의 모델로 SUMO 를 확장할 수 있으며, 시뮬레이션을 원격으로 제어하는데 사용할 수 있는 API 를 제공한다.



[그림 1] SUMO 시뮬레이터

- **RoadRunner**

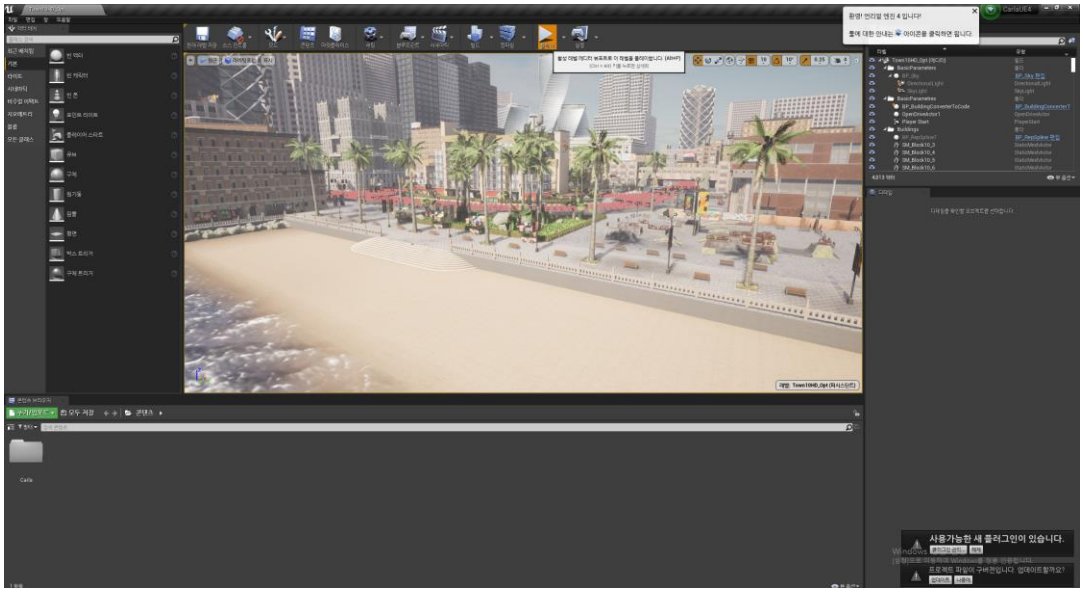
RoadRunner 는 Matlab 에서 제공하는 OpenDRIVE Road Description File 제작을 위한 소프트웨어다. 교통 시뮬레이터에서는 도로 및 교통 신호를 구성하기 위해 OpenDRIVE 라는 파일을 사용하는데, 도로의 구성을 논리적으로 표현하여 여러 교통 시뮬레이터에서 재사용이 가능하다.



[그림 2] RoadRunner 의 교차로

- **CARLA (Car Learning to Act)**

CARLA 는 자율주행 알고리즘의 개발 및 평가를 위한 오픈소스 교통 시뮬레이터로, 다양한 도시 환경과 날씨 조건을 재현하여 사용자들이 편리하게 모델을 테스트할 수 있도록 한다. Unreal Engine 4 를 기반으로 하여 사실적인 그래픽과 물리적 상호작용을 제공하며, 카메라, LIDAR, GPS 등 다양한 센서를 지원하여 정밀한 인식 및 매핑 작업이 가능하다. 고급 트래픽 관리 기능을 통해 주변 차량 및 보행자의 행동을 제어하며, 다양한 데이터 수집 옵션을 제공하여 머신러닝 모델의 훈련 및 검증에 유용하다. 자율주행 관련 연구에서 폭넓게 활용되며, 강력한 Python API 를 제공하여 시뮬레이션 환경을 동적으로 제어할 수 있다.



[그림 3] CARLA 시뮬레이터

3. 연구 내용

3.1. SUMO 시뮬레이터를 활용한 DQN 강화학습

강화학습 모델의 기본 구조를 정의한다. DQN 강화학습에서 사용하는 ReplayBuffer 클래스와, 학습과 분류에 사용될 각종 레이어들을 정의한 QNet 클래스, 주요 여러 멤버변수들을 초기화한 후, 다음 액션을 선택하며 모델의 파라미터들을 업데이트하는 강화학습을 수행하는 DQNAgent 클래스를 정의한다.

SUMO 시뮬레이터 환경을 구성하고, 시뮬레이션의 각 스텝마다 액션을 적용한다. 시뮬레이터 환경은 state 와 reward 를 반환한다. 아래는 강화학습 모델과 연관된 각 메소드의 설명이다.

Step(action)

시뮬레이션의 스텝을 진행하고 강화학습 모델로부터 **action** 을 입력 받아 적용한다. 시뮬레이션의 교차로 신호체계는 총 8 개의 신호단계(**phase**)로 이루어져 있고, 그 중 노란색 신호를 제외하면 유의미한 신호단계는 총 4 개이다. 각 phase 의 길이를 조절하는 것을 **action** 이라고 정했다. 모델이 **traCI** 인터페이스[3]를 이용해 특정 신호단계의 길이를 동적으로 조절한다.

get_state()

각 시뮬레이션 스텝마다 교차로의 상태(**state)**를 나타내는 여러가지 값과 통계치들을 강화학습 모델에 반환한다. traCI 인터페이스를 통해, 시뮬레이션 내부 객체들의 속성 정보를 추출하여 강화학습 모델에 전달한다.

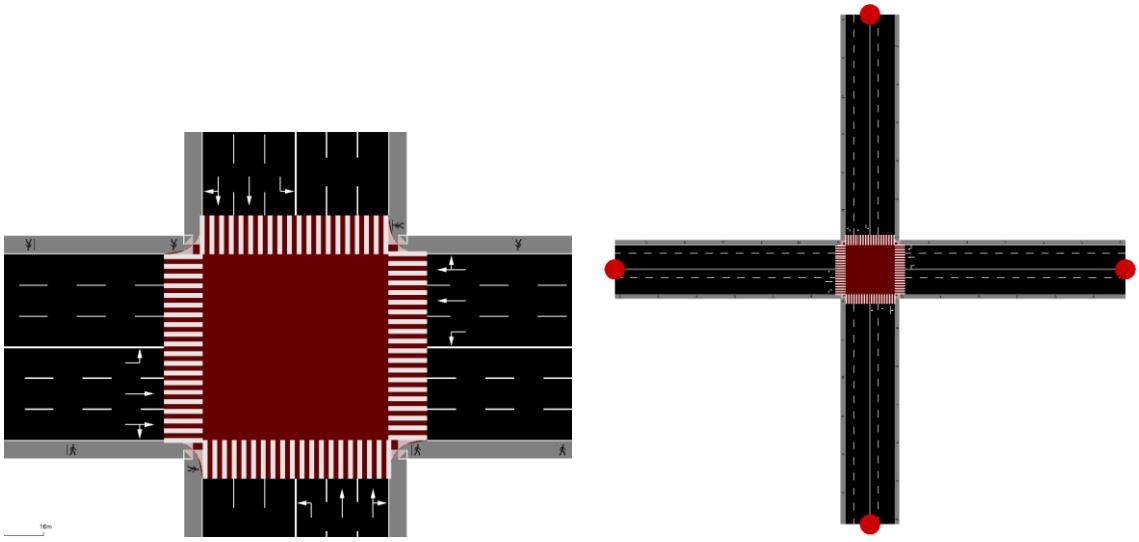
get_reward()

시뮬레이션의 현재 상태에 기반하여 보상(**reward**) 값을 계산한다. **Reward** 는 실시간으로 모든 정지차량의 정지시간 총합, 모든 정지차량의 평균 정지시간, 가장 오래 정지한 차량의 정지시간, 교차로 진입 도로상에 있는 모든 차량 수, 정지한 차량의 총 수 등을 이용해 계산한다. 모든 정지차량의 정지시간 총합에 지수승을 이용하여 reward 를 계산한 후 음수를 취한다. 즉, reward 가 0 에 가까워질수록 우수한 성능을 가진다.

[3] SUMO 시뮬레이터의 Python API

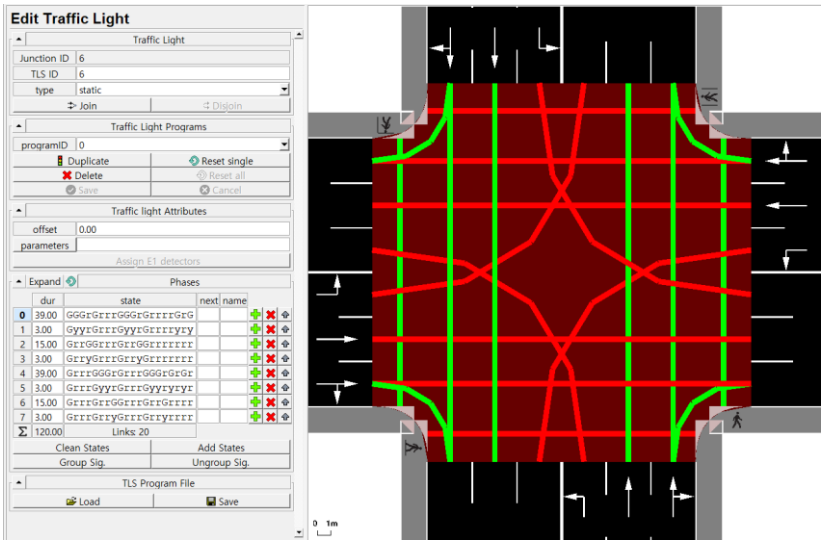
3.1.1. SUMO 교차로 환경 구현

SUMO (Simulation of Urban MObility)는 대규모 교통망을 처리하도록 설계된, 이식성이 뛰어난 오픈소스 멀티모달 교통 시뮬레이션 소프트웨어다. 자동차나 및 보행자 객체들이 특정한 규칙 혹은 무작위로 이동하도록 구현할 수 있고, API 를 통해 정확한 데이터를 얻을 수 있어 초기 설계의 경우, SUMO 의 traCI 를 이용해, 시뮬레이션 데이터를 API 를 이용해 직접 받아 학습하는 것이 모델의 완성도를 높이는데 더 유리하다고 판단했다.



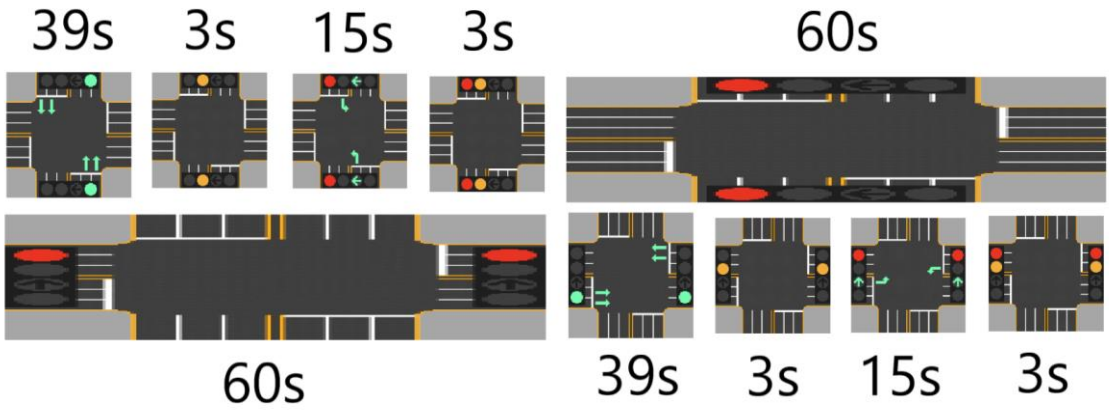
[그림 5] SUMO 로 구현한 교차로

SUMO 시뮬레이터를 구동하기 위해 도로망을 구성하는 각 node, edge, 교차로, 신호등, 횡단보도 등을 .xml 파일로 작성 후, 도로망 위에서 차량들이 이동하는 경로 (route)와 언제, 어디서, 어떤 차량이 생성되고 어느 경로로 이동할 지를 정의한 (vehicle)의 목록을 만든다. 그 다음으로 시뮬레이션 실행에 사용될 네트워크 파일을 제작하고, 시뮬레이션 총 실행 step 등을 기록한다.



[그림 6] 교차로의 connection 및 phase 정보

교차로의 각 방향마다 진입 3 개 차선으로 설계하였다. 교차로에 진입하는 각 차선과 해당 차선에서 향할 수 있는 차선끼리 연결(connection)을 구성한다. 이후에 교차로를 컨트롤하는 신호단계(phase)와 각 신호단계에서 신호등별 점등색과 각 단계의 신호길이(duration)을 정의한다.



[그림 7] 교차로의 phase duration 설계

신호체계를 작성할 때는 위의 그림과 같은 신호단계와 순서, 신호길이를 적용하였다.

마지막으로, 도로망을 실제로 이동할 여러 종류의 차량타입을 정의하고, 교차로를 지나가는 이동경로(route) 별로 무작위로 배치한다.

3.1.2. SUMO API 를 활용한 시뮬레이터 제어

SUMO API 를 사용해 스크립트 내에서 SUMO 시뮬레이를 실행시키고, 실시간으로 교차로의 차량과 신호등의 상태 값을 가져올 수 있다. 또한 주행중인 차량을 개별적으로 컨트롤하거나 신호등의 신호를 제어할 수도 있다. 강화학습 코드를 실행해 시뮬레이터 환경 객체를 생성하고 객체 내에서 시뮬레이터를 제어하고 정보를 얻는다.

3.1.3. SUMO 시뮬레이터와 강화학습 코드 통합

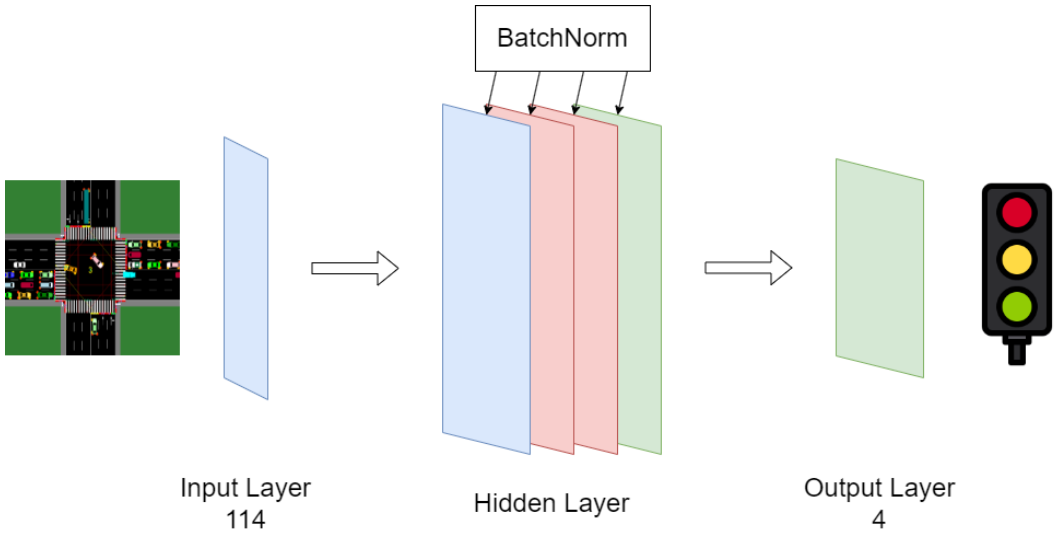
SUMO 의 시뮬레이션 화면은 매우 단순하며 현실과의 괴리가 있다. 따라서 객체인식 모델은 제외하고, 신호 제어 강화학습을 진행하여 교차로 신호를 최적화하는 데에 초점을 두었다. state 를 받아오는 방식은 아래와 같이 2 가지 경우로 시도했다.

1. SUMO API 를 사용, 교차로 상황을 Ground-Truth 숫자 값들로 받아오기 (FCNN)
2. SUMO 의 실시간 시뮬레이션 snapshot 이미지를 받아오기 (CNN)

3.1.3.1. [1] SUMO API 로 받아온 값들을 FCNN 에 입력(이하 'SUMO_APIVALUES _FCNN')

신경망 구성

신경망은 Fully-connected Layer 4 개층을 이용하여 구성했다. 우선 매 Fully-connected layer 를 통과할 때 마다 Batch Normalization 과 Overfitting 을 막기 위한 Dropout 을 적용했다. 그리고, ReLU activation 을 수행하였다. 이런 과정을 동일하게 3 번 반복하고 마지막에 output 을 계산하는 Fully-connected layer 를 통과시켜 액션 4 개 중에 하나를 도출해내도록 했다.



[그림 8] SUMO 강화학습 신경망 구성

GET_STATE 정의

DQN 신경망에 입력할 **state** 는, 현재 시뮬레이션 상태에 관련된 38 개의 값을 3 번 연속적으로 얻는다. 하나의 1 차원 배열을 Input Layer 로 사용하여 강화학습 Agent 에 전달한다.

[현 신호 phase, 현 신호에서 지난 시간,

1 번 lane 상의 전체 차량 수, 정지 차량 수, 평균 속도,

2 번 lane 상의 전체 차량 수, 정지 차량 수, 평균 속도,

3 번 lane 상의 전체 차량 수, 정지 차량 수, 평균 속도,

... (중략) ...

12 번 lane 상의 전체 차량 수, 정지 차량 수, 평균 속도]

[Array] SUMO API 를 사용해 받는 교차로 데이터

연속적인 3 개의 값을 얻는 이유는 DQN 에 현 시뮬레이션 상황에 대한 더 풍부한 정보를 넘겨주기 위함이다. 교차로 상의 각 차량의 위치가 실시간으로 변하고, 순간적인 "**방향성**"을 나타내는 정보를 주는 것이다.

Q-값과 ACTION

Output(Q-value)는 argmax 를 이용하여 출력된 값 중 가장 큰 값에 해당되는 index 를 반환하여 해당 action 을 수행하도록 한다.

REWARD 계산

Reward 는 DQN 신경망을 학습시키는데 있어서 가장 중요한 부분으로, 신경망에서 출력한 action 을 현재 state 에 적용시켜 일정 시간 후, 다음 state 를 얻는다. 해당 시뮬레이션 환경에서 계산한 reward 가 반환되면, DQN 신경망은 이것이 최대화될 수 있는 정책 즉 **'action pattern'**을 학습한다. 이 프로젝트의 경우, 대기시간을 최소화 하는 것을 목표로 설정했다.

$$reward_{step} = \frac{-모든차량의총대기시간}{모든대기중인차량의수}$$

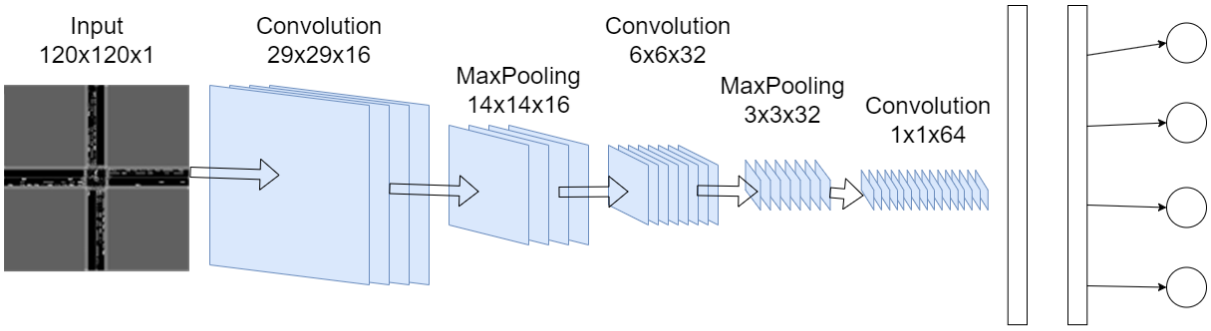
[수식 2] Reward 계산식

위와 같이 reward 를 계산하고, 대기시간을 최소화하여 reward 를 최대화 하는 것을 목표로 설정했다.

3.1.3.2. [2] SUMO 의 시뮬레이션 snapshot 이미지를 CNN 에 입력(이하 'SUMO_IMG_CNN')

신경망 구성

두 번째 방식은 1 차원 배열이 아닌 2 차원 흑백 이미지를 처리하기 위해 **CNN(Convolutional Neural Network)**을 사용한다. 신경망은 3 개의 Convolutional Layer 와 마지막에 2 개의 Fully-connected Layer 를 이용하여 구성하였다. 각 Convolution 층 사이에는 Max Pooling 을 수행한다. 이후 나온 결과를 1 차원 배열로 만들어 Fully-Connected Layer 를 이용해 액션을 도출했다.



[그림 9] SUMO 강화학습 신경망 구성

GET_STATE 정의

DQN 신경망에 입력할 **state** 는, 현재 시뮬레이션의 이미지를 Grayscale 로 적용하여 120x120 크기의 흑백 이미지로 강화학습 Agent 에 전달한다. Q-Value 와 Reward 는 이전 방법과 동일하게 계산했다.

3.2. CARLA 시뮬레이터를 활용한 DQN 강화학습

3.2.1. CARLA 교차로 환경 구현

시뮬레이션 환경을 서버로, 모델측 동작을 클라이언트로 구성하고, Python API 를 이용해, 서버와 클라이언트가 더 유동적이고 상호적으로 동작할 수 있도록 설계했다.

시뮬레이션 환경

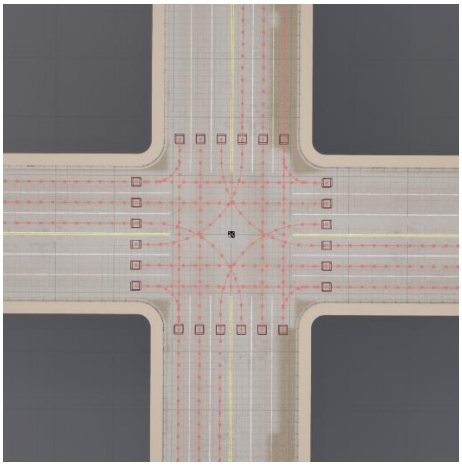
구상한 시뮬레이션 환경은 3 차선의 도로가 4 갈래로 나뉘는 총 12 개의 차선이 있는 교차로이다. 각 차선에는 차량 오브젝트가 Spawn 되는 구간이 있고, 해당 차량은 반대편 도로 끝에 도달하면 오브젝트는 Destroy 되도록 구성했다.

[그림 10] CARLA 시뮬레이터를 이용한 교차로 시뮬레이션

모델의 성능에 관한 논의 결과, 실제 도로에서도 사용할 수 있게 만들기 위해서는 시뮬레이션 환경이 충분한 다양성을 갖춰야 한다는 결론을 내렸다.

먼저, 차량이 교차로에 도달하는 시간의 무작위성을 부여하기 위해, 각 차선에서 차량 오브젝트가 Spawn 되는 주기는 최소 50 tick 에서 최대 300 tick 중 무작위로 선택되고, 이 시간 이후에 일정 확률로 Spawn 위치에 생성된다. 생성 이후에는 또 다시 50 tick 에서 최대 300 tick 중 무작위 시간 뒤에 생성이 될 가능성이 생긴다. 또한 특정 구간에서의 생성될 확률을 너 높이고, 줄이는 것을 가능하게 해서 특정 구간에만 차량이 몰리는 상황도 구현이 가능하다.

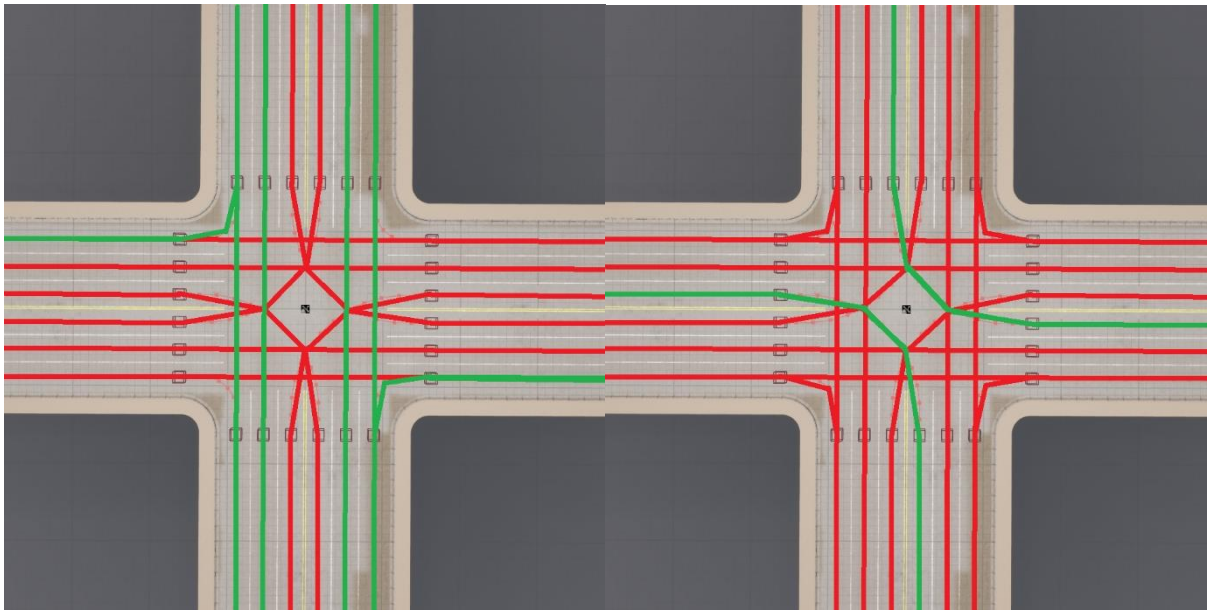
차량의 진행 경로는, 1 차선 차량은 좌회전 전용 차선, 2 차선과 3 차선은 직진이 가능하고, 3 차선은 우회전도 가능하다. 3 차선의 차량은 직진을 할지, 우회전을 할지 차량 오브젝트마다 무작위 선택된다. 이를 나타내면 아래 그림과 같이 나타낼 수 있다.



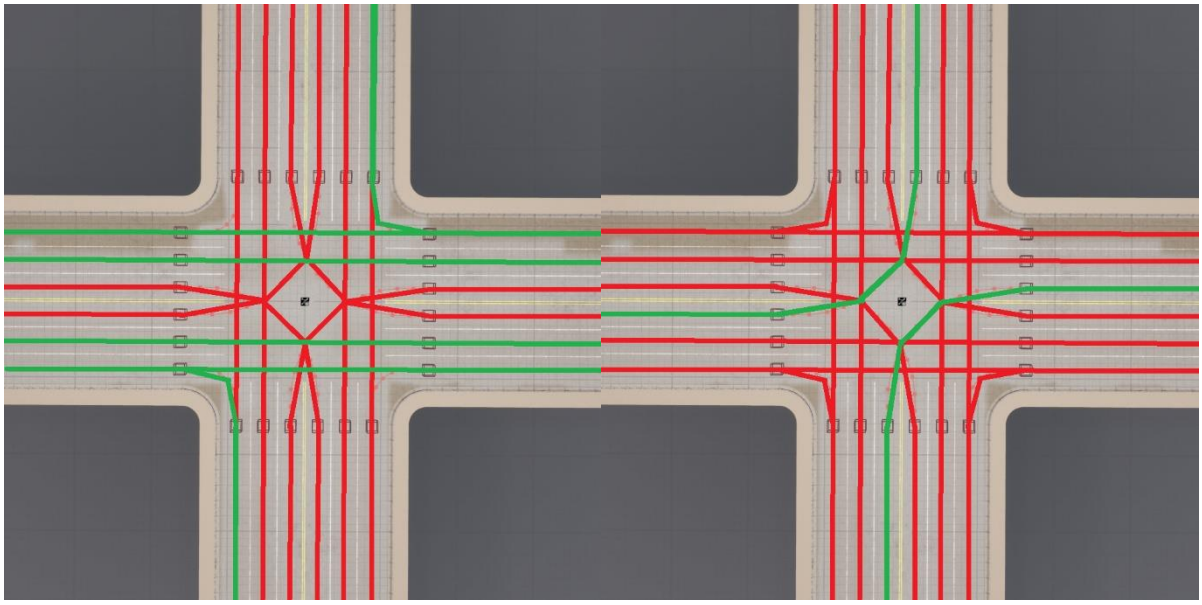
[그림 11] 차선별 이동 가능 경로

모델 인터페이스

모델이 스스로 시뮬레이션 환경을 조작 가능하도록 총 4 가지 Action 을 구성했다. Action 0 은 남 북 방향 도로의 직진과 우회전 신호. Action 1 은 남 북 방향 도로의 좌회전 신호, Action 2 는 동 서 방향 도로의 직진과 우회전 신호. Action 3 은 동 서 방향 도로의 좌회전 신호이다.



[그림 12] Action 0 과 Action 1 의 신호 제어



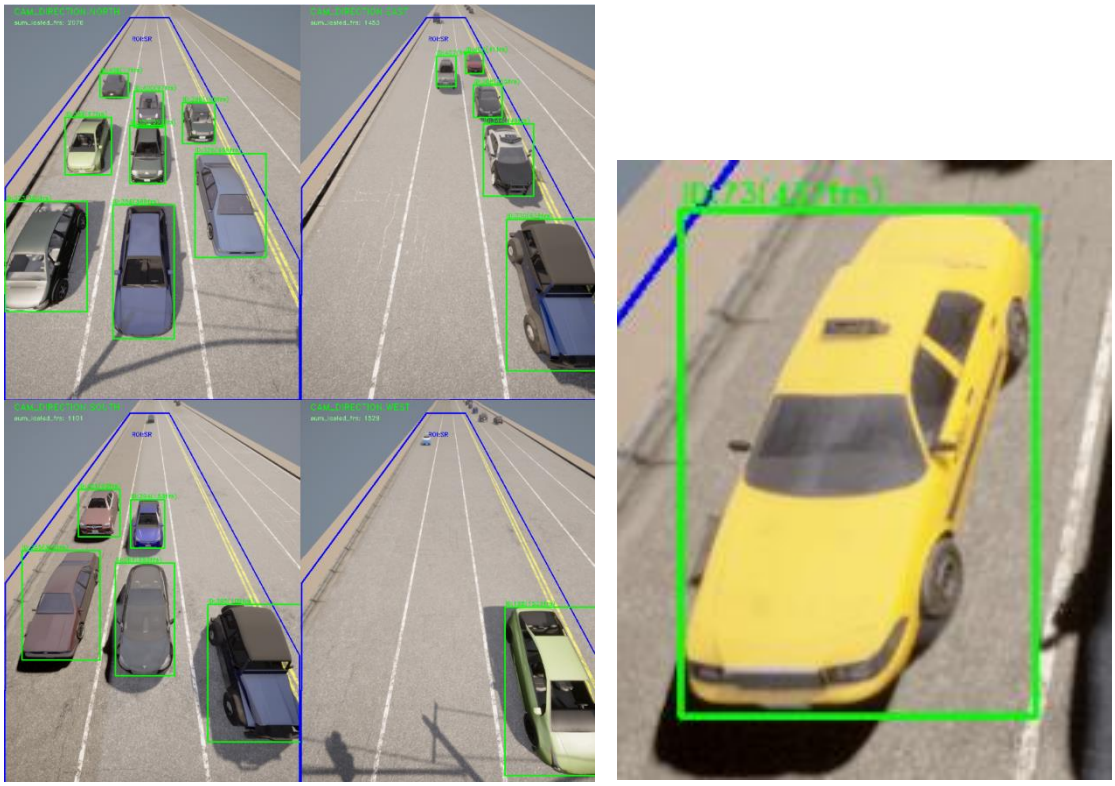
[그림 13] Action 2 과 Action 3 의 신호 제어

3.2.2. CARLA Python API 를 활용한 시뮬레이터 제어

먼저 CARLA 의 환경 객체를 생성하여 시뮬레이션 서버에 접속하여 동기식 연결을 하고, 시뮬레이션 환경을 설정하고, 관심구역에 실시간 이미지를 입력 받기 위한 카메라를 설치한다. 시뮬레이션을 진행하기 위해 start(), step(action), reset() 세 가지 메소드를 정의하는데, **start()**는 episode 가 시작되기 위한 환경을 구성하고 제어할 신호등의 Actor 객체를 얻는다. **step(action)**은 Agent 의 요구에 따라 각 신호단계에 맞게 신호등을 제어한다. **reset()**은 지금까지 생성된 오브젝트들을 메모리 상에서 제거하고, 다음 episode 를 진행시키기 위한 준비를 한다.

3.2.3. CARLA 시뮬레이터와 강화학습 코드 통합

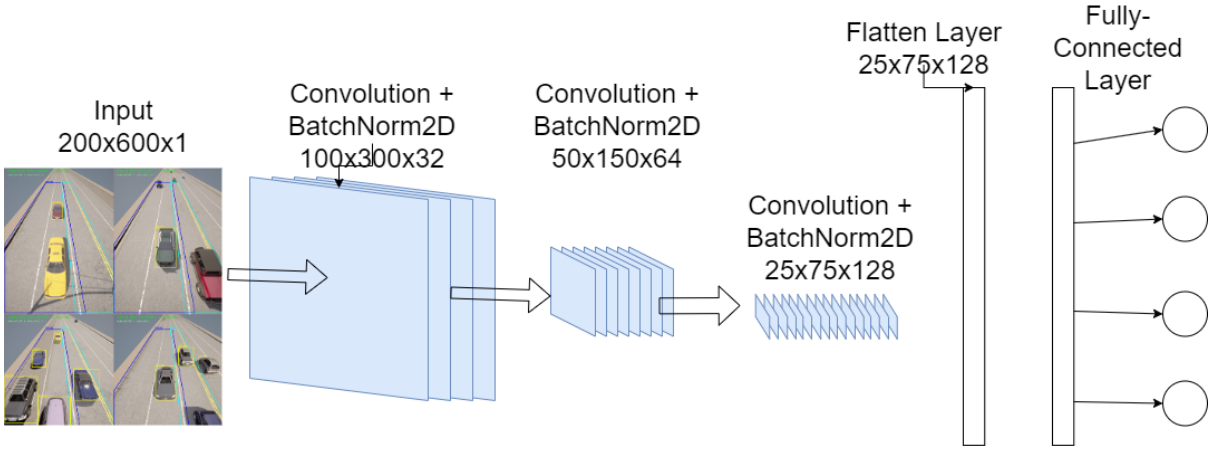
CARLA 시뮬레이터는 교차로와 차량의 모습을 현실과 유사한 이미지로 렌더링하여 보여준다. 설정한 각 카메라로부터 이미지를 받아, state 를 얻고, reward 를 계산하기위해 YOLO 와 ByteTrack 로 각 개체가 대기한 시간 t와 도로 방면별 대기중인 차량의 수를 측정한다. 즉, CARLA 를 활용한 강화학습은, 현실의 교차로에 카메라를 설치하여 교차로 최적화 시스템을 구축하기 위한 예비 실험이며 우리 프로젝트의 최종 목표이다.



[그림 14] 각 도로에서의 객체 탐지 및 추적, 대기시간 측정

신경망 구성

각 방면의 도로에서 200 x 150 x 1 크기의 Grayscale 이미지를 입력 이미지로 사용한다. 신경망은 Convolution Layer 과 BatchNorm2D 층을 3 개 층을 통과시킨 후, Flatten 시켜, Fully-Connected Layer 에 통과시켜 4 개의 Action 중 하나를 도출한다.



[그림 15] CARLA 강화학습 신경망 구성

REWARD 계산

CARLA 시뮬레이터 사용의 목표는 제공되는 API 를 사용하지 않고, YOLO, ByteTrack 모델을 이용하여 직접 Reward 를 계산하는 것이다. 네 방면의 교차로 정지선 앞, 관심 구역(ROI: Region of Interest)으로 설정한 구역에 들어온 차량이 추적된 프레임 수를 계산한다. 자동차와 트럭, 버스 등을 탐지하고 위치정보를 이용해 Bounding Box 를 그릴 수도 있다.

$$reward = -(NS_{SR} + NS_L + EW_{SR} + EW_L)$$

[수식 3] Reward 계산식

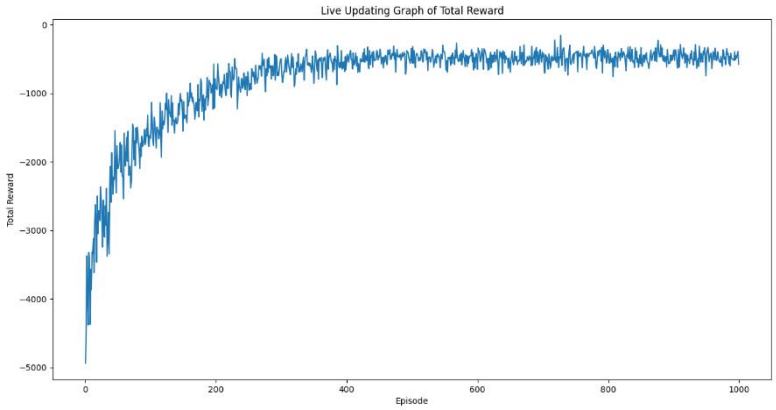
NS_{SR}, NS_L, EW_{SR}, EW_L 에서 NS는 남, 북 도로, EW는 동, 서 도로를 말한다. SR은 직진, 우회전 신호, L은 좌회전 신호를 말한다. 차량의 수, 대기시간t를 기반으로 해당 reward 를 계산한다.

4. 연구 결과 분석 및 평가

4.1. SUMO 모델 학습 및 테스트 결과

4.1.1. 'SUMO_APIVALUES_FCNN'의 경우

- **모델 학습 그래프**



[그림 16] SUMO 강화학습 모델 1 번 – Reward 그래프

에피소드당 스텝수	2,000
sync_interval	20
epsilon_start	0.9
epsilon_end	0.1
epsilon_decay	0.995
gamma	0.98
lr	0.001
buffer_size	50,000
batch_size	64
진행 에피소드	0~1,000

[표 1] CUMO 강화학습 모델 1 번 – 파라미터

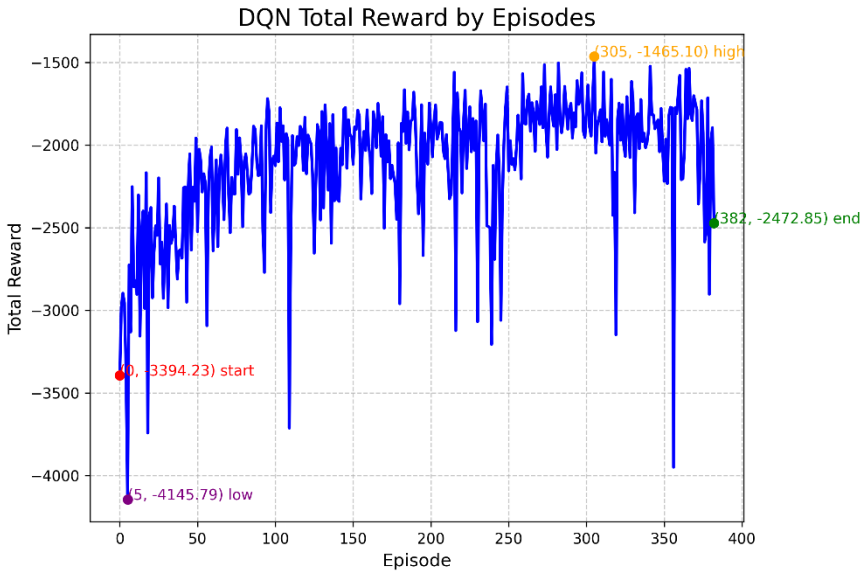
- **결과**

Total reward 는 -5,000 부근에서 시작하여 조금씩 진동하지만 매우 분명하고 또렷하게 상승하였다. 처음부터 끝까지 모든 에피소드가 지나는 동안 total reward 값이 크게 아래로 떨어지는 경우가 없었고 하락폭은 작았다. 전체적으로 깔끔하게 상승하는 그래프가 나타났다. 더불어, 학습을 진행하는 동안 연산속도가

매우 빠르다는 것을 느꼈는데, 아마 레이어 수가 적고 간단한점, STATE 으로 넘겨주는 것이 길이가 길지 않은 1 차원 배열이라 그런 것으로 여겨진다.

4.1.2. 'SUMO_IMG_CNN'의 경우

- **모델 학습 그래프**



[그림 17] SUMO 강화학습 모델 2 번 – Reward 그래프

에피소드당 스텝수	2,000
sync_interval	20
epsilon_start	0.9
epsilon_end	0.1
epsilon_decay	0.995
gamma	0.98
lr	0.001
buffer_size	10,000
batch_size	32
진행 에피소드	0~382

[표 2] CUMO 강화학습 모델 2 번 – 파라미터

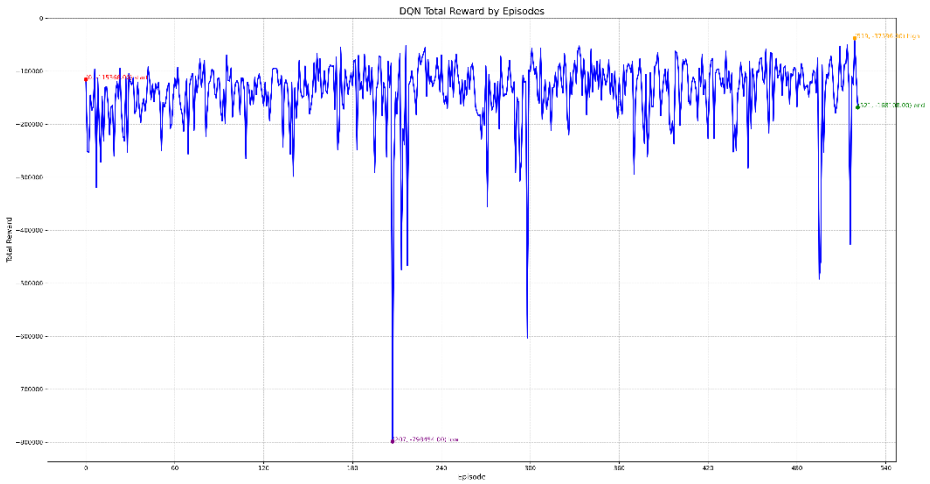
초반부의 reward 값은 -3,000 정도에서 시작했다가, 학습이 진행될수록 평균적인 reward 값이 증가하였다. 후반부에 가까워질수록 평균적인 reward 값은 -1,500 ~ -2,000 사이의 값이 되었고, 학습 전과 비교했을 때, 유의미한 차이를 보였다.

4.2. CARLA 모델 학습 및 테스트 결과

아래 3 가지 모델의 Q-Network 구조는 모두 동일하며 이전에 서술한 바와 같다. 다만 카를라 환경 설정 상의 미세한 변화와 한 에피소드당 스텝수, 학습과 관련한 하이퍼파라미터들을 조금씩 변화시키 실험하였다.

4.2.1. 모델 1 번

- **모델 학습 그래프(Total Reward by Episodes)**



[그림 18] CARLA 강화학습 모델 1 번 – Reward 그래프

에피소드당 스텝수	2,000
sync_interval	20
epsilon_start	0.9
epsilon_end	0.1
epsilon_decay	0.99
gamma	0.9
lr	0.0005
buffer_size	20,000
batch_size	32
진행 에피소드	0~512

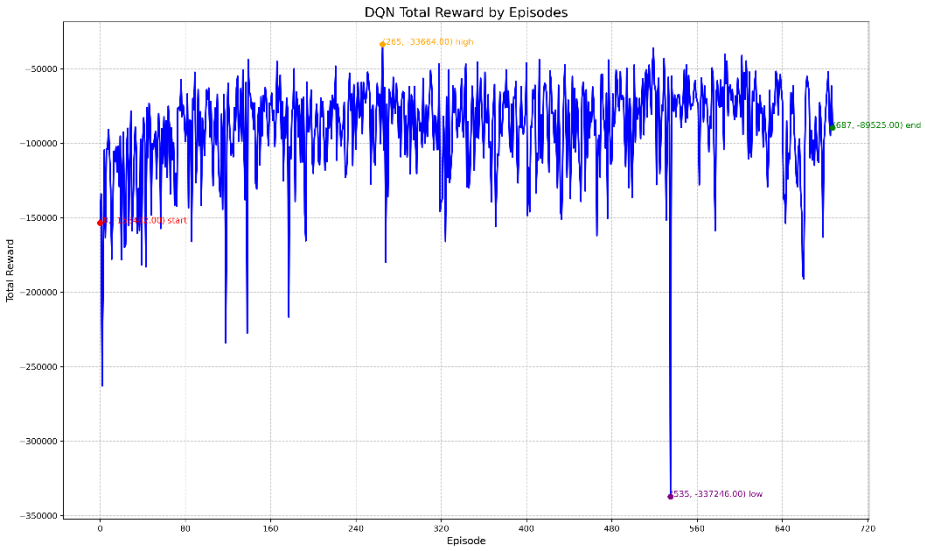
[표 3] CARLA 강화학습 모델 1 번 – 파라미터

- **결과**

Total reward 는 -75,000 에서 -200,000 사이를 요동치는 결과가 나왔다. 최고값은 -37,596 까지도 나왔지만, 시작값인 -110,000 보다 낮은 약 521 에피소드에 와서는 -170,000 으로 오히려 시작값보다 너 낮은 리워드가 나왔다. 또한 꾸준히 상승하지도, 하락하지도 않으며 수렴하지 않았다.

4.2.2. 모델 2 번

- **모델 학습 그래프(Total Reward by Episodes)**



[그림 19] CARLA 강화학습 모델 2 번 – Reward 그래프

에피소드당 스텝수	2,000
sync_interval	20
epsilon_start	0.9
epsilon_end	0.1
epsilon_decay	0.995
gamma	0.98
lr	0.0001
buffer_size	20,000
batch_size	32
진행 에피소드	0~687

[표 4] CARLA 강화학습 모델 2 번 – 파라미터

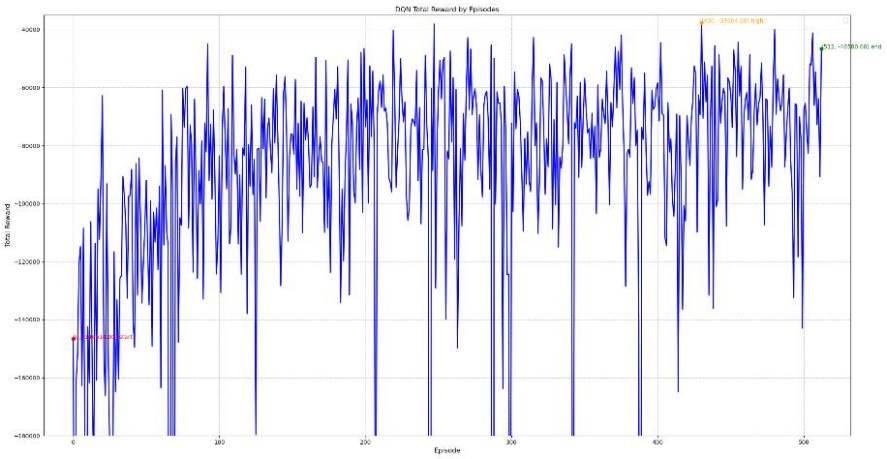
- **결과**

Epsilon decay 를 0.995 로 적용해 이전보다 더 천천히 감소하도록 하여, 더 많은 시간동안 에이전트가 환경을 탐색할수 있도록 했다. gamma 는 0.98 로 이전보다 높여서 장기적인 보상이 중요하고 더 복잡한 정책을 잘 학습하도록 하였다.

이전 학습에서는 값이 너무 큰 폭으로 움직여 불안정했기 때문에, Learning Rate 를 0.0001 로 낮추어 학습 정밀도를 높이고자 했다. 이전 학습보다 값들의 진동폭이 좀 작아졌지고, total reward 가 아주 미세하게나마 경향성을 가지고 조금씩 향상되는 결과가 나왔다.

4.2.3. 모델 3 번

- **모델 학습 그래프(Total Reward by Episodes)**



[그림 20] CARLA 강화학습 모델 3 번 – Reward 그래프

에피소드당 스텝수	3,000
sync_interval	20
epsilon_start	0.9
epsilon_end	0.1
epsilon_decay	0.995
gamma	0.98
lr	0.0001
buffer_size	20,000
batch_size	32
진행 에피소드	0~512

[표 5] CARLA 강화학습 모델 3 번 - 파라미터

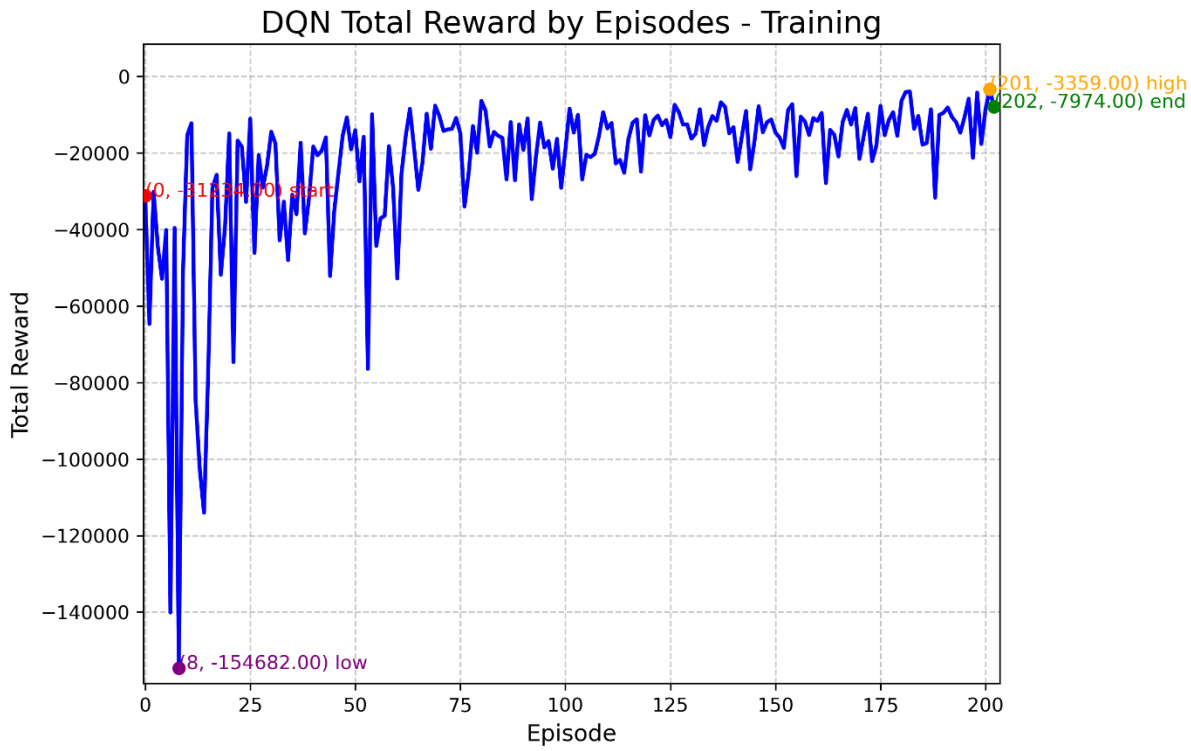
- **결과**

세 번째 모델은, 에피소드당 스텝의 수를 3,000 스텝으로 증가시켰다. 기존에는 교차로에 차량이 몰리고 신호가 바뀌어 차량이 통과하는 과정이 짧아 에이전트가 학습한 시간이 충분하지 않다고 판단했다.

이전 학습들보다 한결 나은 결과가 나타났다. 중간에 값이 크게 떨어지는 부분들은 확인한 결과, 시뮬레이터 상의 오류로, 차량들이 서로 걸려, 움직이지 못하게 된 상황에서의 reward 값이다. 값이 요동치는 폭은 아직 작다고 할수 없지만 이번에는 분명히 서서히 상승하는 흐름을 볼수 있었다. 초반의 -120,000 ~ -160,000 구간에서 꾸준히 상승하여 -40,000 ~ -80,000 구간까지 상승하였다. 512 에피소드에서 모델학습을 종료하고 모델 테스트를 진행하였다.

4.2.4. 모델 4 번

- **모델 학습 그래프(Total Reward by Episodes)**



[그림 21] CARLA 강화학습 모델 4 번 – Reward 그래프

에피소드당 스텝수	3,000
sync_interval	20
epsilon_start	0.9
epsilon_end	0.1
epsilon_decay	0.995
gamma	0.98
lr	0.0001
buffer_size	20,000
batch_size	32
진행 에피소드	0~202

[표 6] CARLA 강화학습 모델 4 번 - 파라미터

- **결과**

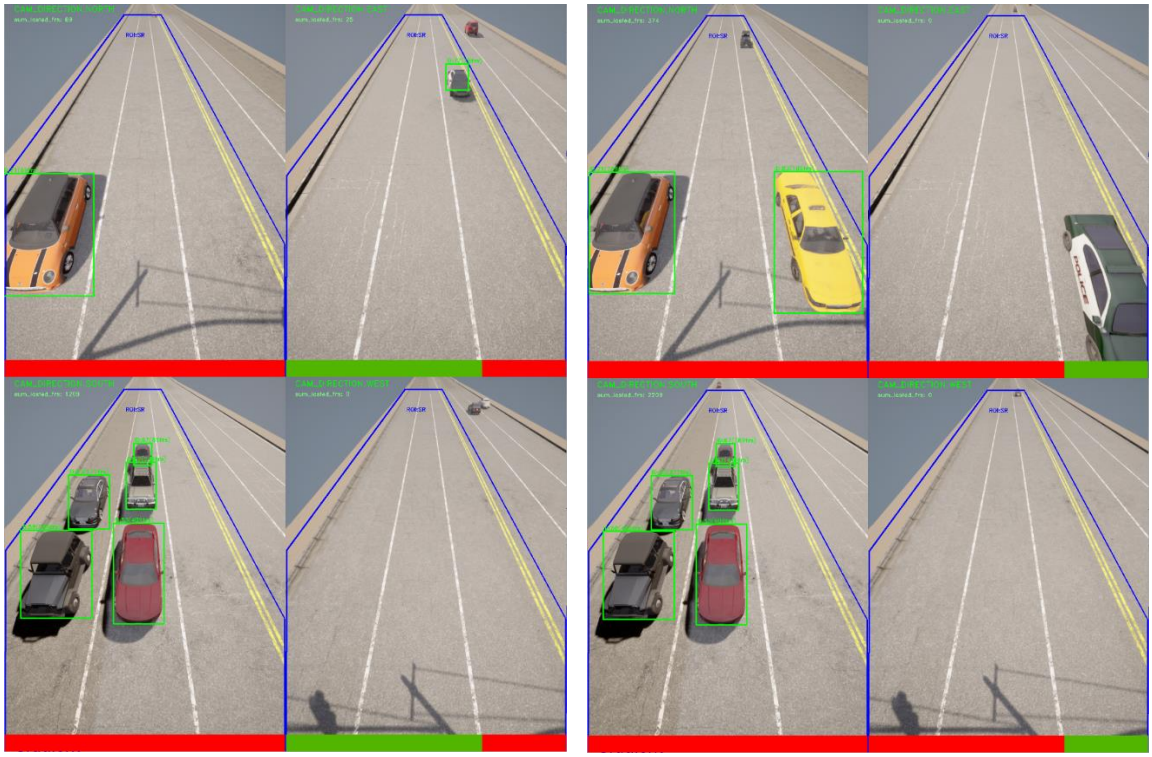
이전 모델에서는 차량들이 모든 방면에서 무작위 주기와 확률로 생성되었다. 하지만, 모든 방향에서 무작위의 교통량이 발생하면 강화학습 모델이 최적화를 했음에도, 고정주기식 신호 형태와 비슷한 결과를 낼 것이다. 따라서 네 번째 모델은 이전과 달리, 일정 주기마다 8 개의 경로중 하나를 무작위로 선택하고, 해당 방향의 교통량이 급증하도록 설계하였다. 이런 환경에서는 모델이 상황을 명확하게 분별하고 학습을 더욱 쉽게 할 것이다.

실제로 이 환경에서 학습한 결과, reward 그래프는 이전과 다른 결과를 나타냈는데, 값의 상방은 점진적으로 상승하였고, 진폭 크기가 크게 줄어들었다. 초반부의 reward 의 상방은 약 -30,000 에서 시작하여 후반부에서는 약 -3,000 까지 상승했다. 모델의 추론 성능이 200 에피소드 부근에서 안정적으로 변하는 것으로 보여 학습을 종료하였다.

4.3. 기존 고정 신호 제어와 강화학습 신호 제어의 비교

아래의 내용들은 모델 4 번으로 추론하여 테스트를 진행한 내용이다. 현실의 교차로의 고정주기식 신호 제어를 재현한 경우(1)와 강화학습 모델이 신호를 추론하도록 한 경우(2)를 비교하였다.

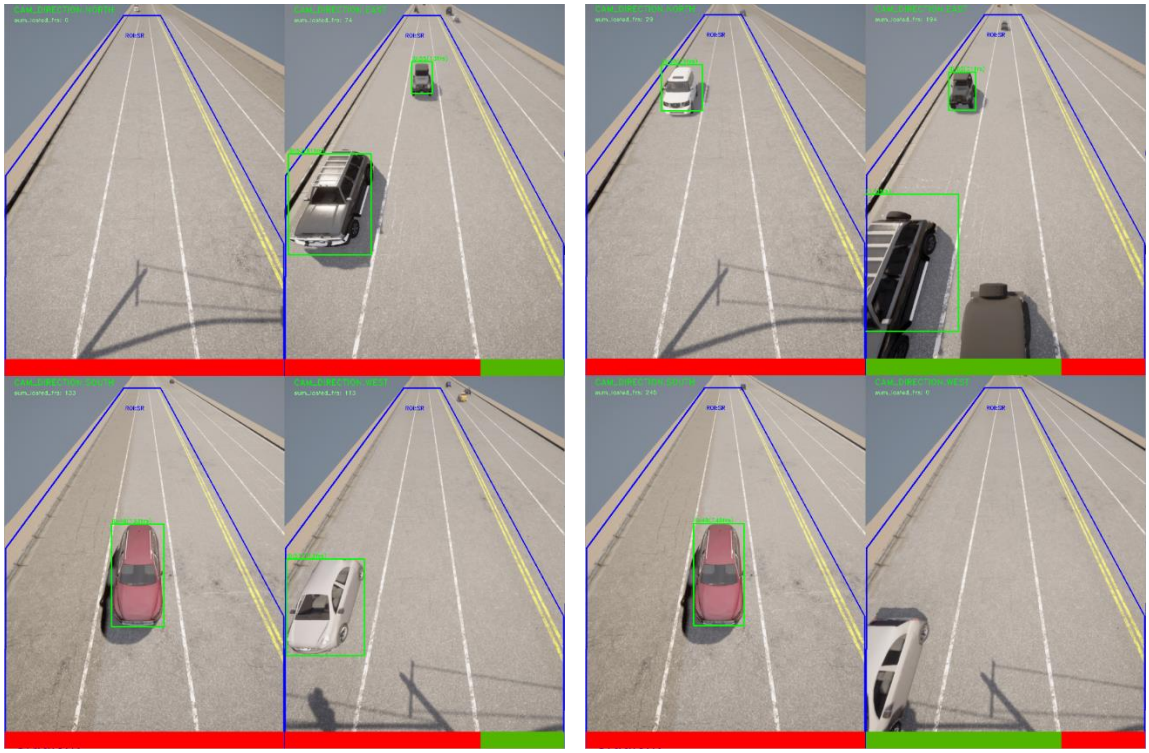
(1)기존 고정 신호 제어



[그림 22] 고정 신호 제어의 현재 신호(좌)와 다음 신호(우)

기존 시간고정식 신호 제어 방식에서는, 현재 남,북 방향 도로의 직우 차선에 대기중인 차량이 많은 상황임에도 불구하고, 고정적으로 정해진 신호에 따라, 동,서 방향 도로의 좌회전 차선에 녹색 신호를 주어 비효율적이다.

(2)강화학습 신호 제어



[그림 23] 강화학습을 이용한 동적 신호 제어의 현재 신호(좌)와 다음 신호(우)

강화학습을 거친 동적 신호 제어의 경우, 시간고정식 제어와 다르게, 차량의 수와 대기 시간을 반영한다. 위 그림의 경우, 동,서 방향 도로의 직우 차선에 대기중인 차량과 진입중인 차량을 인식하고, 고정식 신호 제어의 경우, 남,북 도로의 직우 차선에 신호를 줄 차례이지만, 동적 신호 제어의 경우, 순서와 관계없이 해당 차선에 녹색 신호를 주어 효율적인 교통 제어를 보여준다.

이상의 실험들과 결과에서, 우리가 학습시켜 완성한 강화학습 모델이, 교통량이 예측 불가하게 증감하는 상황에서 능동적으로 매 상황에 맞추어 교통량이 많은 방면에 신호를 합리적으로 준다는 것을 확인하였다.

5. 결론 및 향후 연구 방향

결론

이 프로젝트에서는 강화학습 기반의 동적 교차로 신호 제어 시스템을 개발했다. Q-Net 의 CNN 구조를 활용하여 교차로의 다양한 이미지를 입력 데이터로 사용하였다. 하지만, GPU 메모리의 한계로 인해 신경망의 구조를 단순하게 설계하고 Batch 크기를 늘리기 어려워 효율적인 학습에 제약이 있었다. 더욱 정밀한 실시간 상황 판단을 위해 단일 프레임이 아닌 연속된 프레임을 입력으로 사용하고, 다양한 시뮬레이션 환경에서 학습을 진행함으로써 모델의 일반화 능력을 높일 수 있을 것이다.

학습 과정을 진행하면서, total reward 가 급격히 상승하며 수렴하는 모습을 뚜렷하게 보지는 못했지만, 실시간 시뮬레이션 화면을 통해 모델의 판단이 발전하는 과정을 지켜볼 수 있었다. 초기 단계에서는 모델의 결정이 내가 내리는 판단과 다소 차이가 있었으나, 학습 후반에 접어들면서 점차 모델의 판단이 인간의 판단과 유사해지는 경험을 하였다. 이는 강화학습 모델이 실제 교차로 상황에 적응하고, 교통 흐름을 효과적으로 제어하기 위해 학습하고 있다는 긍정적인 신호로 여겨진다. 이러한 경험은 모델이 점진적으로 교차로의 교통 패턴을 이해하고, 더 나은 신호 제어 결정을 내리는 데 기여하고 있음을 시사한다.

향후 연구 방향

다양한 강화학습 알고리즘: DQN 외에도 PPO 와 A3C 와 같은 효율적인 알고리즘을 적용할 수 있다. 특히 PPO 는 연속된 state-action 공간에서 안정적이고 빠르게 수렴하여, 실시간 교차로 동적 신호 제어에 더욱 적합하다. 또한, 모델 기반 강화학습을 통해 교차로의 차량 제어가 환경 변화에 더 민감하게 대응할 수 있을 것이다.

시뮬레이션 환경의 다양성: 현재 특정 교차로에 대한 시뮬레이션에 한정되지 않고, 다양한 시나리오를 통해 보다 일반화된 모델을 개발한다. 예를 들어, 도심 뿐만 아니라 고속도로, 좁은 교차로 등에서 학습을 진행하고, 보행자, 자전거 등 다양한 요소를 고려한 시뮬레이션 환경을 구축해야 한다.

다중 에이전트 강화학습 (Multi-Agent Reinforcement Learning): 여러 교차로 간의 협력을 통해 전체 교통 흐름을 최적화할 수 있는 시스템을 개발할 수 있다. 다중 에이전트 시스템을 적용하여 각 교차로가 상호작용하며 학습할 수 있도록 하여, 자율적이고 협력적인 신호 제어가 가능해질 것이다.

6. 참고 문헌

[1] Hyunjin Joo, & Yujin Lim (2020). Distributed Traffic Signal Control at Multiple Intersections Based on Reinforcement Learning. The Journal of Korean Institute of Communications and Information Sciences, 45(2), 303-310, 10.7840/kics.2020.45.2.303 (in Korean)
[2] Jawoon Gu, Minhyuck Lee, & Chulmin Jun (2020-11-20). Traffic signal control simulation based on reinforcement learning. Proceedings of Korean Society for Geospatial Information Science, Seoul. (in Korean)