서강대학교 SOGANG UNIVERSITY

강화학습을 통한 교통 흐름 최적화 및 자율 주행 신호등 제어

데이터 사이언스 · 인공지능

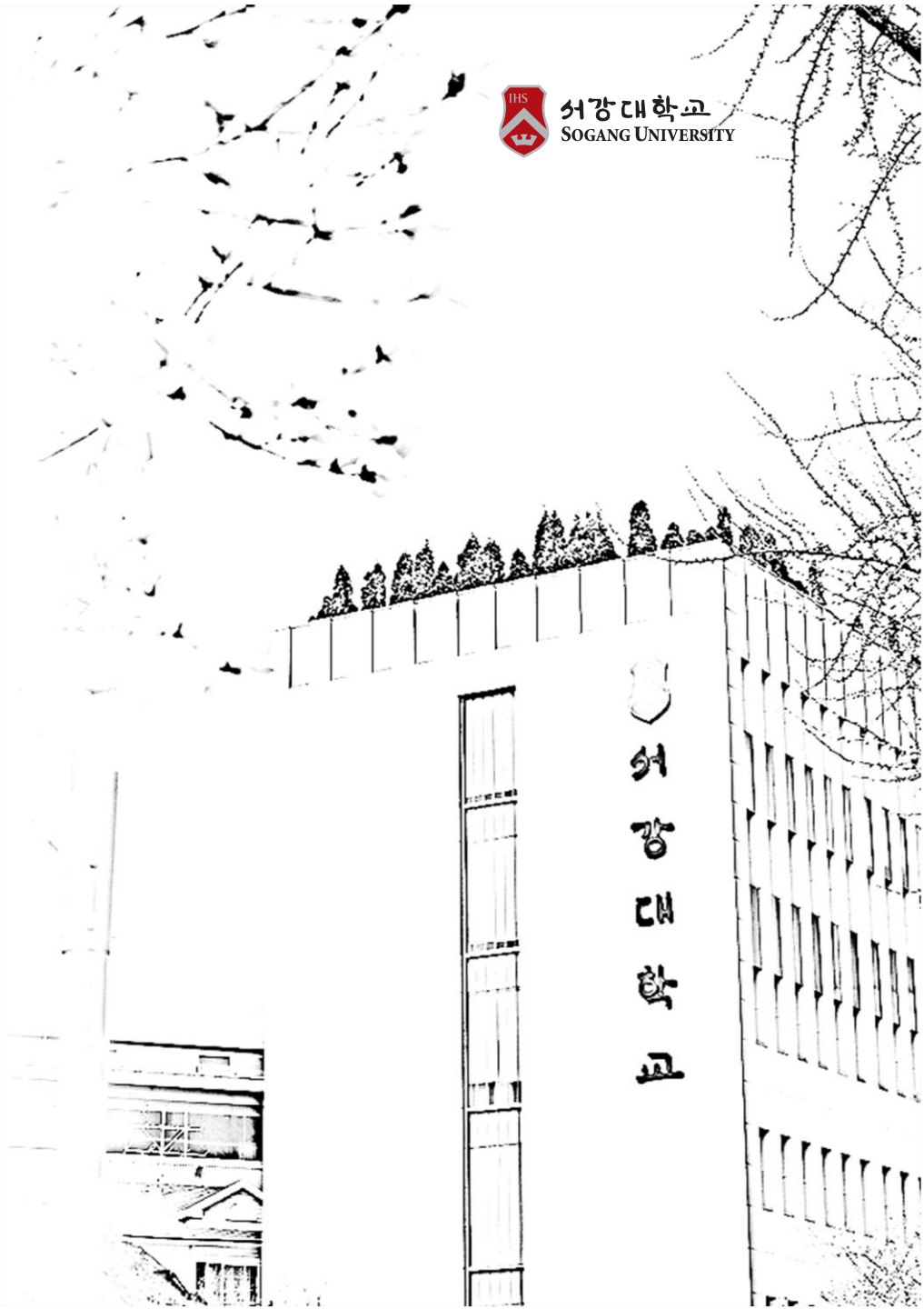
이민규 (A70074)
오현택 (A70067)

저장소
https://github.com/hy1as/traffic_control_reinforce

CONTENTS

SOGANG UNIVERSITY

00. 팀원 기여 사항



00. 팀원 기여 사항

서강대학교 SOGANG UNIVERSITY

0.1 기여 사항

이 민 규 (A 7 0 0 7 4)

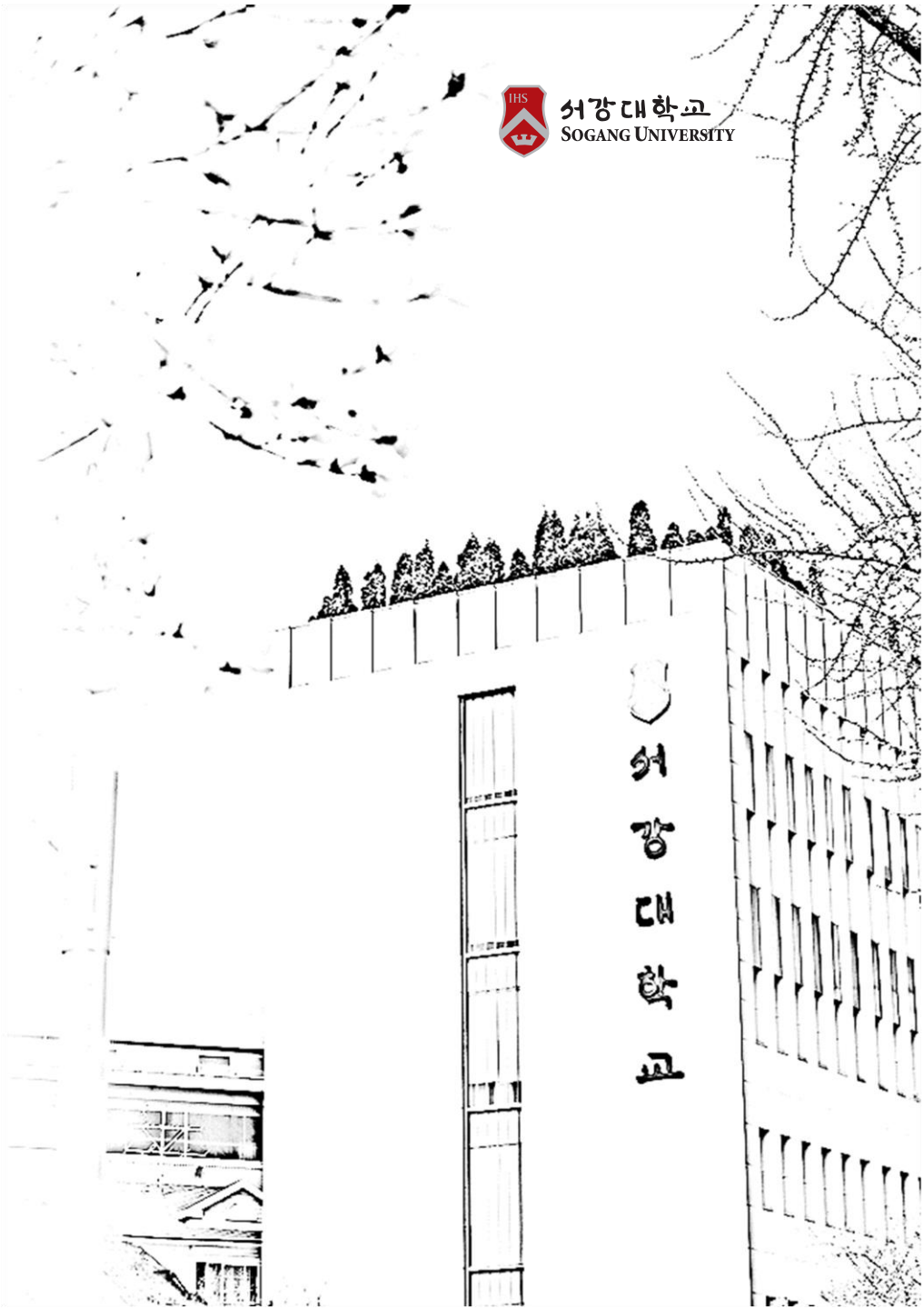
- Action/Reward 설계 (신호 변경, 대기시간 패널티 등)
- 가상 데이터셋 생성 로직 설계 및 구현
- 강화학습 알고리즘 구현 (DQN, DDQN 등)
- 하이퍼파라미터 튜닝 및 비교 실험
- 보고서 및 GitHub README 작성

오 현 택 (A 7 0 0 6 7)

- 4거리 교차로 환경(시뮬레이터) 구현
- State 설계 (차량 대기열, 신호 상태 등)
- 실험 결과 시각화 (학습 및 대기시간 & 처리량 그래프)
- 신호 처리 시각화 시뮬레이션 구현
- 보고서 작성

SOGANG UNIVERSITY

01. 프로젝트 주제 및 목표



01. 프로젝트 주제 및 목표

서강대학교 SOGANG UNIVERSITY

1.1 프로젝트 배경

▶ 도심 교통 혼잡의 심각성

- 전 세계 주요 도시에서 교통 체증으로 인한 경제적 손실은 연간 수조 원에 달하며, 운전자들의 평균 통근 시간이 지속적으로 증가
- 특히 교차로는 교통 흐름의 병목 구간으로, 비효율적인 신호 체계는 전체 도로망의 소통에 직접적인 영향

▶ 기존 신호등 시스템의 한계

- 대부분의 교차로는 **고정된 주기**로 작동하여, 실시간 교통량 변화에 유연하게 대응 불가
- 새벽 시간에는 차량이 거의 없는데도 적색 신호를 대기하여야 하고, 출퇴근 시간에는 한쪽 방향에만 교통량이 치중되어도 균등하게 신호 배분하는 비효율 발생

01. 프로젝트 주제 및 목표

서강대학교 SOGANG UNIVERSITY

1.2 프로젝트 주제

주 제	**"강화학습을 통한 교통 흐름 최적화 및 자율 주행 신호등 제어"**
핵심 아이디어	교차로를 Environment로 모델링하고, 신호등을 Agent로 설정하여, 차량 대기 시간을 최소화하는 정책 학습
차별점	기존 고정 주기 신호등 대비 동적 상황 대응력을 실험적으로 비교 검증

기대효과

- 사회적 효과: 교통 체증 완화, 연료 소비 및 탄소 배출 감소
- 학술적 기여: 강화학습 기반 교통 제어 시스템의 구현 사례 제공

01. 프로젝트 주제 및 목표

서강대학교 SOGANG UNIVERSITY

1.3 프로젝트 목표

▶ 기술적 목표

- 사거리 교차로 환경 시뮬레이션 구현
- 교통 시스템에 맞는 MDP 설계
- 강화 학습 알고리즘 적용하여 Agent 학습
- 정량적 지표(평균 대기 시간, 처리 차량 수 등)을 통한 모델 성능 측정

▶ 실험 목표

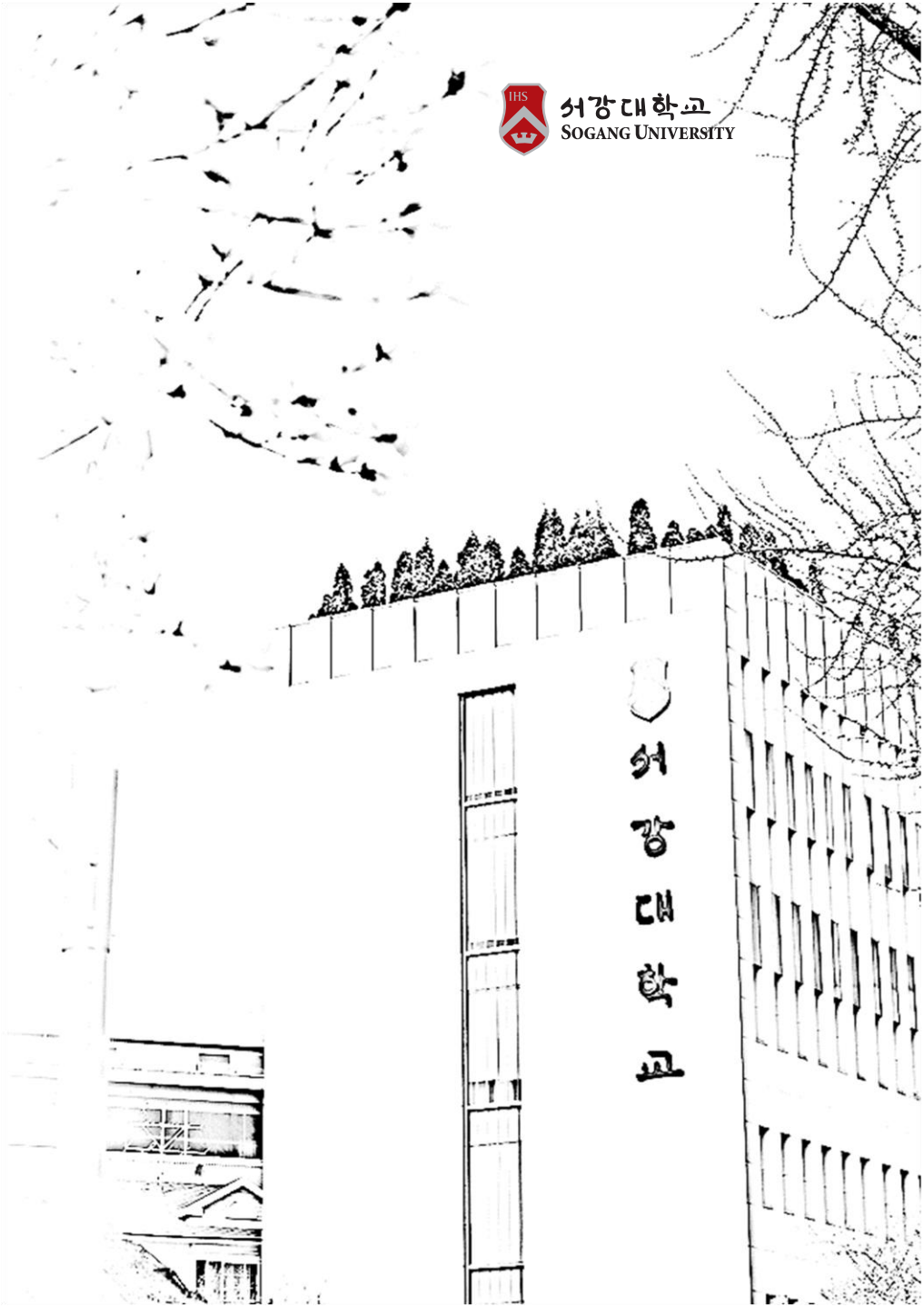
- **Baseline 비교**: 고정 주기 신호등과의 성능 비교
- **시나리오 테스트**: 출퇴근 시간대와 같이 특정 방향으로 차량이 집중되는 상황에서의 강화학습 모델의 적응력 검증
- **하이퍼파라미터 분석**: Learning rate, Reward Function 변화에 따른 학습 효율 비교

▶ 학술적 목표

- 실제 교통 데이터를 참조하여 현실성 있는 시뮬레이션 환경 구축
- 프로젝트 결과를 통해 강화학습의 실용적 응용 가능성을 보고서와 시각화 자료로 제시

SOGANG UNIVERSITY

02. 환경 및 데이터셋 설명



02. 환경 및 데이터셋 설명

서강대학교 SOGANG UNIVERSITY

2.1 교차로 환경 구현

▶ 환경 구조

모델	동(east), 서(west), 남(south), 북(north) 4개 방향 진입 차선 구성

- 클래스 기반 교차로 직접 구현
- 각 차선은 Queue 자료구조로 관리되며, 차량의 진입과 통과를 시뮬레이션
- GUI 대신 데이터 구조만으로 환경을 표현하여 학습 속도 최적화

▶ 제약 사항

- 차량 통과 시간: 1초당 1대 (현실적인 처리 속도)
- 노란불 시간: 2초 (도로교통법 기준)
- 최소 신호 유지 시간: 5초 (안전성 고려)

▶ 환경 동작 메커니즘

항목	내용
시간 단위	1 step = 1초
차량 생성	매 스텝마다 확률적으로 각 차선에 차량이 진입
	Poisson Distribution을 활용한 차량 도착 패턴 구현
차량 이동 규칙	초록불: 해당 차선의 맨 앞 차량이 교차로를 통과하고 큐에서 제거
	빨간불: 차량은 대기하며 대기 시간 누적
신호 전환 제약	신호 변경 시 노란불 페이즈(2초) 강제 적용으로 빈번한 신호 교체 방지

02. 환경 및 데이터셋 설명

서강대학교 SOGANG UNIVERSITY

2.2 데이터셋 구성

수집 전략	실제 데이터 기반 가상 생성 방식 채택

▶ 가상 교통 데이터

- 수집한 실제 통계치를 시뮬레이터의 차량 도착 확률로 변환
- Poisson Distribution을 통한 확률적 차량생성

```python
import numpy as np


# 예시: 오전 출근 시간대 북쪽 차선 차량 생성
arrival_rate = 0.6  # 초당 0.6대 (실제 데이터 기반)
num_vehicles = np.random.poisson(arrival_rate)
```

시나리오	시간대	북/남 차선	동/서 차선	비고
평시	낮 시간	0.2대/초	0.2대/초	균등한 교통량
출근 시간	오전 8~9시	0.6대/초	0.15대/초	북→남 집중
퇴근 시간	오후 6~7시	0.15대/초	0.6대/초	동→서 집중
혼잡 상황	이벤트 발생	0.8대/초	0.7대/초	전방향 과부하

02. 환경 및 데이터셋 설명

서강대학교 SOGANG UNIVERSITY

2.2 데이터 전처리 및 정규화

▶ 상태 정보 정규화

- 각 차선의 대기 차량 수를 0~1 범위로 스케일링

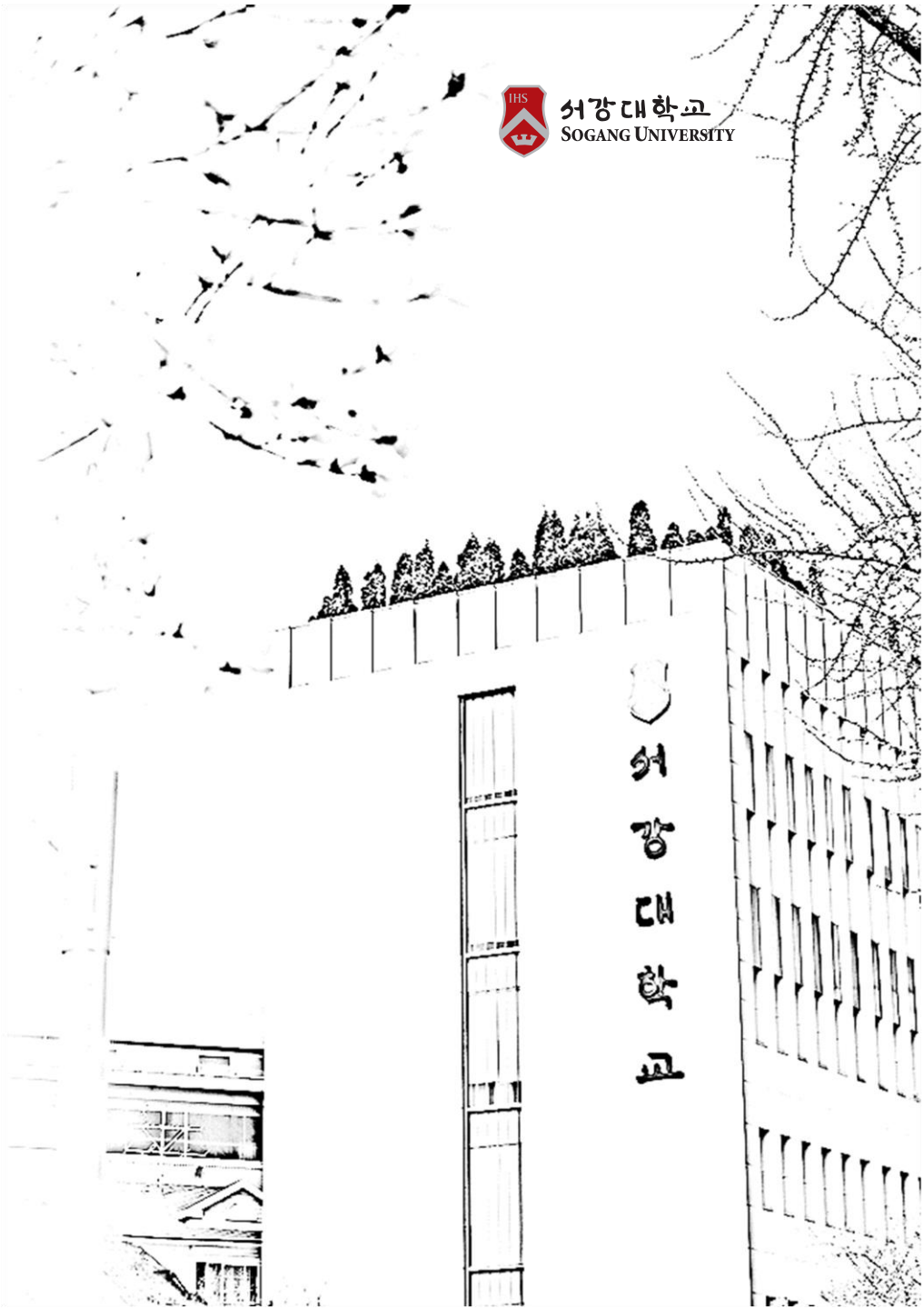
```
normalized_queue = min(queue_length / MAX_QUEUE_LENGTH, 1.0)
```

▶ 시간 정보 인코딩

- 현재 신호의 지속 시간을 State에 포함하여, Agent가 신호 변경 타이밍을 학습할 수 있도록 설계
- 시간대(오전/오후/야간) 정보를 One-hot Encoding하여 시나리오별 패턴 학습 지원

SOGANG UNIVERSITY

03. MDP 설계



03. MDP 설계

서강대학교 SOGANG UNIVERSITY

3.1 State 설계

▶ State 공간 구성

정의 신호등(Agent)가 매 스텝 관찰할 정보

- 북쪽 차선 대기 차량 수
- 남쪽 차선 대기 차량 수
- 동쪽 차선 대기 차량 수
- 서쪽 차선 대기 차량 수
- 현재 신호 상태 (0: 북/남 초록, 1: 동/서 초록)
- 현재 신호 지속 시간 (초)
- 시간대 정보 (0: 야간, 1: 평시, 2: 출퇴근)

```
State = [
    queue_north,      # 북쪽 차선 대기 차량 수
    queue_south,      # 남쪽 차선 대기 차량 수
    queue_east,       # 동쪽 차선 대기 차량 수
    queue_west,       # 서쪽 차선 대기 차량 수
    current_phase,    # 현재 신호 상태 (0: 북/남 초록, 1: 동/서 초록)
    phase_duration,   # 현재 신호 지속 시간 (초)
    time_of_day       # 시간대 정보 (0: 야간, 1: 평시, 2: 출퇴근)
]
```

- 마주 보는 신호 (북/남, 동/서)는 실제 환경을 참고해 하나의 신호로 판단

03. MDP 설계

서강대학교 SOGANG UNIVERSITY

3.1 State 설계

▶ State 표현 방식

방식	이산형 상태(원본)	연속형 상태(정규화)
정의	대기 차량 수를 정확한 숫자로 표현 예시) [5, 12, 3, 8, 0]	대기 차량 수를 구간으로 나누어 표현 예시) 0~5대: 0 (원활) , 6~10대: 1 (보통) 등
장점	정보 손실 없음	상태 공간 축소로 학습 효율 증가
단점	상태 공간이 넓어 학습이 어려울 수 있음	일부 정보 손실
채택		○

▶ 정규화 (대기 차량 수)

```python
def normalize_state(state):
    """상태를 0~1 범위로 정규화"""
    max_queue = 30  # 최대 대기 차량 수
    max_duration = 60  # 최대 신호 지속 시간

    normalized = [
        min(state[0] / max_queue, 1.0),  # 북쪽 큐
        min(state[1] / max_queue, 1.0),  # 남쪽 큐
        min(state[2] / max_queue, 1.0),  # 동쪽 큐
        min(state[3] / max_queue, 1.0),  # 서쪽 큐
        state[4],                        # 신호 상태 (0 or 1)
        state[5] / max_duration,         # 신호 지속 시간
        state[6] / 2.0                   # 시간대 (0~2 → 0~1)
    ]
    return normalized
```

서강대학교 SOGANG UNIVERSITY

3.2 Action 설계

▶ Action 공간 정의

정의	신호등(Agent)가 매 스텝마다 선택할 수 있는 행동

- 0 : 현재 신호 유지
- 1 : 신호 변경 (북/남 ↔ 동/서)

▶ Action 제약 조건

Action	설명	효과
신호 유지	신호 변경 시 2초 노란불 페이즈 강제 삽입	실제 환경에서의 안전 고려
신호 변경	신호를 켠 후 최소 5초 이상 유지해야 변경 가능	초기 학습 단계에서 Agent의 이상 행동(차 이동 없이 신호만 반복 변경) 방지

03. MDP 설계

서강대학교 SOGANG UNIVERSITY

3.3 Reward 설계

▶ Reward Function의 목표

최적화 목표	교차로에 있는 모든 차량의 총 대기 시간 최소화

▶ Reward Function 설계

보상 요소	수식	의도
기본 페널티	$-\times$ 총 대기 차량	대기 차량 수 최소화 하도록 유도
신호 변경 비용	-5	이상 행동(차량 이동 없이 신호만 변경) 방지
혼잡 가중 페널티	$-($ 특정 방향 대기차량 $-\ 15) \times 2$	특정 방향 과부화 방지
처리 보너스	$+1 \times$ 통과 차량	차량 처리 장려

▶ Reward Function 구현

```python
def reward(state, action):
    """
    음수 보상: 대기 차량이 많을수록 큰 페널티
    목표: 보상을 0에 가깝게 만들기 (= 대기 차량 최소화)
    """
    total_waiting = (state['queue_north'] +
                     state['queue_south'] +
                     state['queue_east'] +
                     state['queue_west'])

    # 기본 페널티: 대기 차량 수에 비례
    reward = -1.0 * total_waiting

    # 추가 페널티: 신호 변경 시 (빈번한 변경 방지)
    if action == 1:  # 신호 변경
        reward -= 5.0  # 변경 비용

    # 추가 페널티: 대기 시간이 긴 차량에 가중치
    for queue in [state['queue_north'], state['queue_south'],
                  state['queue_east'], state['queue_west']]:
        if queue > 15:  # 혼잡 임계값
            reward -= (queue - 15) * 2.0  # 추가 페널티

    return reward
```

03. MDP 설계

서강대학교 SOGANG UNIVERSITY

3.4 Transition 설계

▶ Transition 정의

정의	교차로 환경이 한 스텝(1초) 진행되는 과정

▶ 1 Step 당 Transition 과정

1 단계 현재 교차로 State 관찰

↓

2 단계 신호등 Action 선택 (유지 or 변경)

↓

3 단계 신호 제어 (노란불 처리 포함)

4 단계 신규 차량 진입 (확률적 생성)

↓

5 단계 차량 통과 (초록불 방향만)

↓

6 단계 대기 시간 누적

↓

7 단계 보상 계산 및 상태 업데이트

↓

8 단계 종료 조건 확인

↓

9 단계 결과 저장 및 반환

03. MDP 설계

서강대학교 SOGANG UNIVERSITY

3.5 Discount Factor 설계

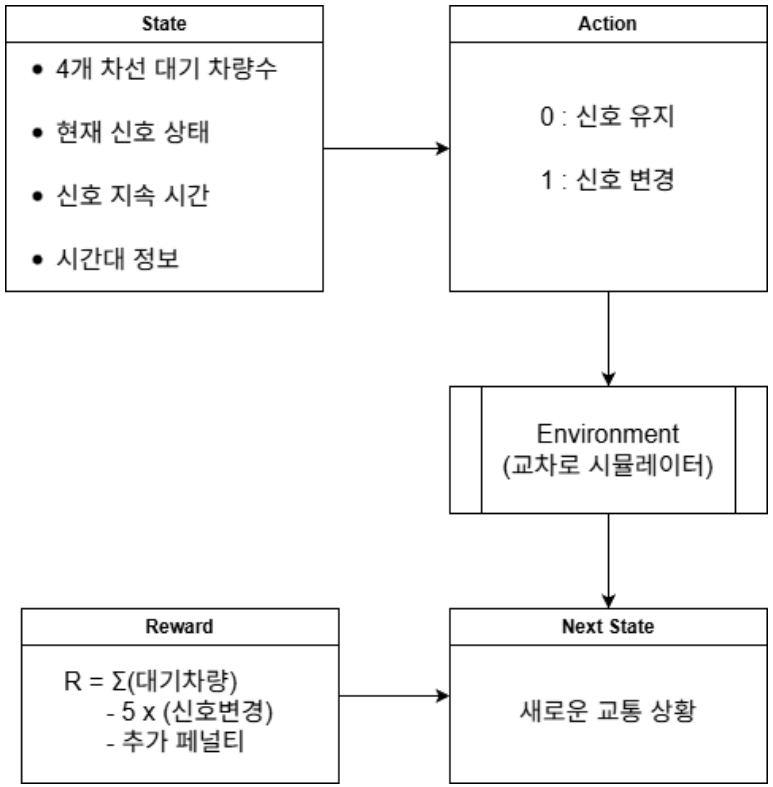
▶ Discount Factor 설정

설정	$\gamma = 0.9 / 0.95 / 0.99$

3.6 MDP 설계 요약

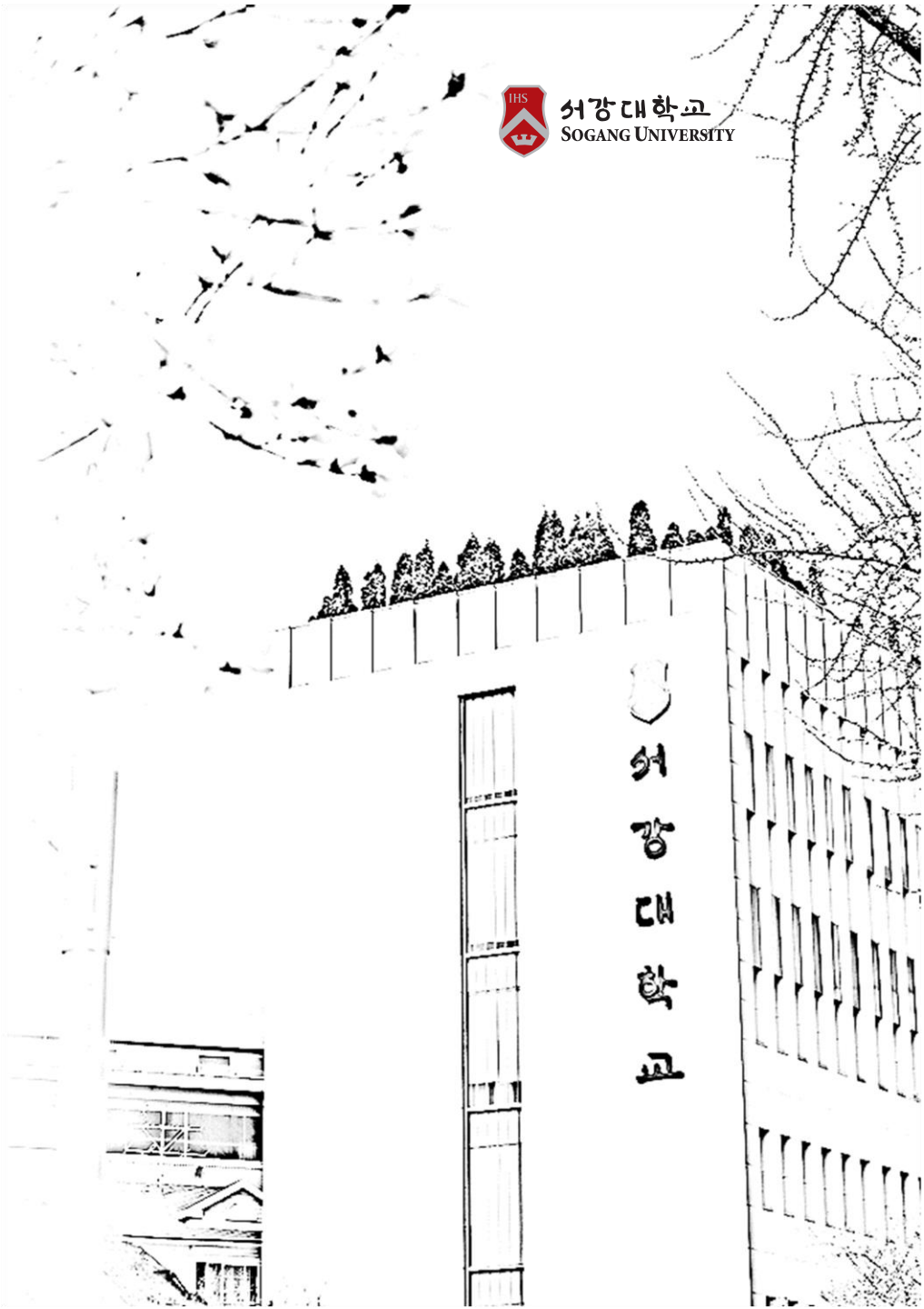
요소	요약
State	7차원 vector 로 교차로 state 표현
Action	학습 효율성 위한 2-action 단순화
Reward	대기 시간 최소화 + 안정성 의 균형
제약	노란불, 최소 유지 시간 반영 등 현실 세계 고려

▶ MDP 시각화



SOGANG UNIVERSITY

04. 알고리즘 설계



04. 알고리즘 설계

서강대학교 SOGANG UNIVERSITY

4.1 알고리즘 선정

▶ 알고리즘 비교 분석

알고리즘	방법론	Action 공간	특징
Q-Learning	Value-based	Discrete	표 형태 Q-Value
DQN	Value-based	Discrete	신경망 Q-Function 근사
REINFORCE (Policy Gradient)	Policy-based	Discrete/ Continuous	Monte Carlo, 높은 분산
DDPG	Policy-based	Continuous	Continuous Action
PPO	Policy-based	Discrete/ Continuous	안정적

▶ DQN 선정 이유 (1)

1 Discrete Action 최적화

- 신호등 제어는 명확한 2가지 Action(유지/변경)의 선택지 존재
- DQN은 각 행동의 Q-Value 직접 출력
- Policy Gradient는 불필요한 복잡도 추가

2 함수 근사 능력

- Neural Network 통한 무한에 가까운 State space 처리
- Q-Learning 방식의 한계 해결

04. 알고리즘 설계

서강대학교 SOGANG UNIVERSITY

4.1 알고리즘 선정

▶ DQN 선정 이유 (2)

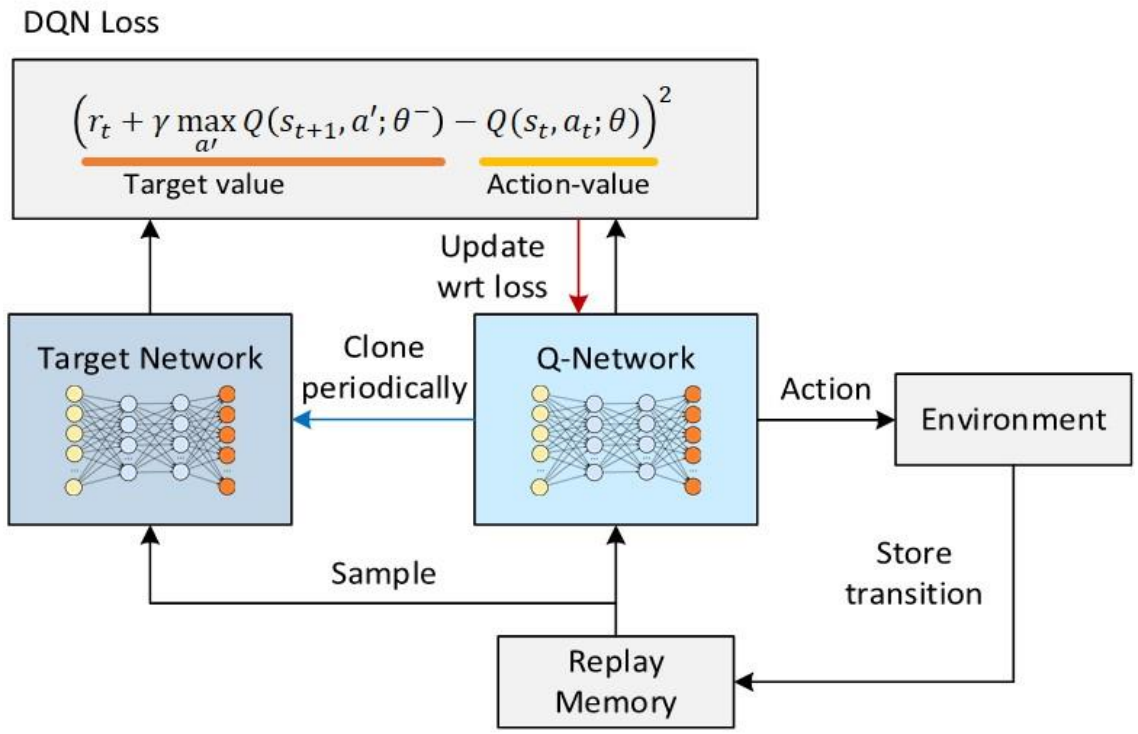
3 안정적 학습

- Experience Replay: 과거 경험 재사용으로 학습 안정화
- Target Network: 목표값 고정으로 수렴성 향상
- REINFORCE 대비 낮은 분산

4 데이터 효율성

- Off-policy 학습: 과거 데이터 재사용
- 같은 경험을 여러 번 학습 가능

▶ DQN 알고리즘 시각화

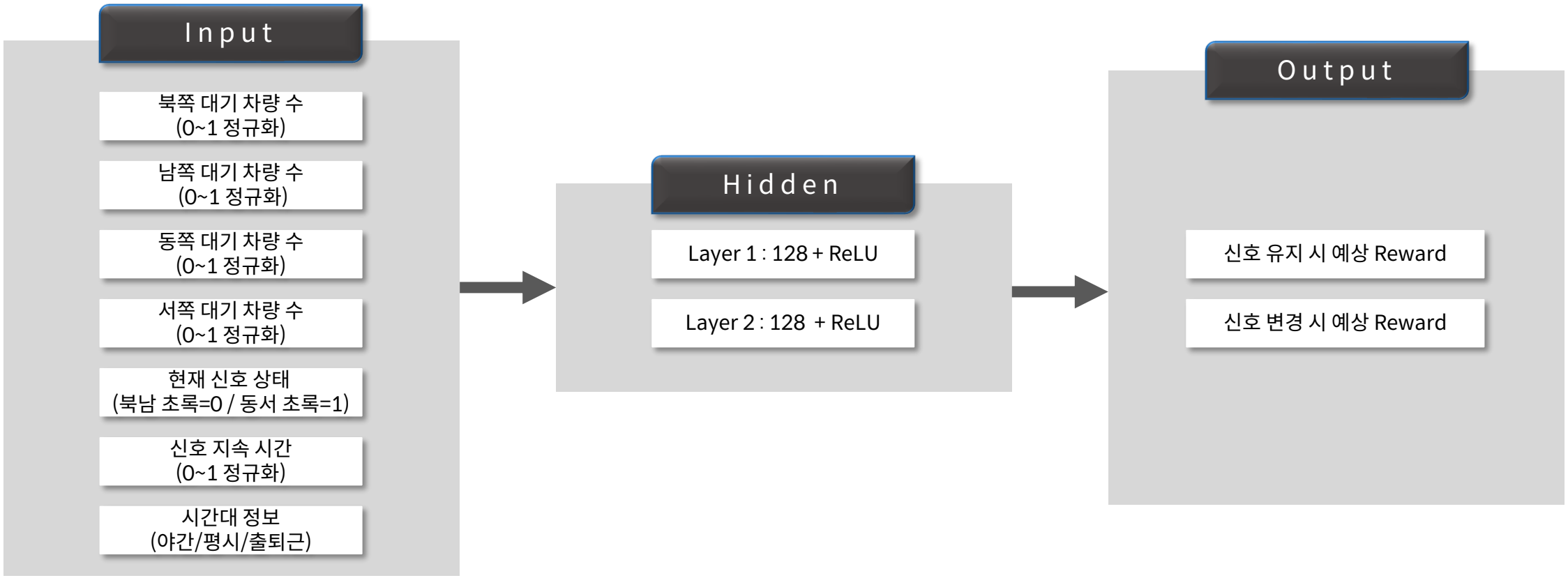


04. 알고리즘 설계

서강대학교 SOGANG UNIVERSITY

4.2 DQN 알고리즘 설계

▶ 네트워크 구조 설계



서강대학교 SOGANG UNIVERSITY

4.2 DQN 알고리즘 설계

▶ DQN 핵심 메커니즘

1 Experience Replay Buffer

- 용량: 10,000개 경험
- 샘플링: Batch size - 64
- 목적: 경험 간 상관관계 제거, 데이터 재사용, 다양한 상황 경험

3 ε-Greedy 탐색

- 초기값: ε = 1.0 (100% 탐색)
- 최종값: ε = 0.01 (1% 탐색)
- 감소율: ε = ε × 0.995 (매 Episode)
- 목적 : 다양한 상황 및 학습된 정책 활용

4 Target Network

- 업데이트 주기: 100 Episode
- 방식: Main Network의 가중치 복사
- 목적: 학습 목표값 안정화, 발산 방지

04. 알고리즘 설계

서강대학교 SOGANG UNIVERSITY

4.3 하이퍼파라미터 설정

▶ 기본 설정

종류	값	비고
Learning rate	α = 0.001	Adam optimizer 사용
Discount Factor	γ = 0.95	약 20 Step 이동 고려
Batch Size	B = 64	학습 안정성과 속도의 균형
Relay Buffer	Capacity = 10,000 Min Size = 1,000	

종류	값	비고
탐색 전략	ε_start = 1.0 ε_end = 0.01 ε_decay = 0.995	
Target Network	Update Frequency = 100 episodes	
학습 설정	Episodes = 2,000 Max Steps = 1,000 (per episode)	

04. 알고리즘 설계

서강대학교 SOGANG UNIVERSITY

4.3 하이퍼파라미터 설정

▶ 튜닝 계획

1 Learning Rate

- 학습 속도와 안정성의 균형
- α = 0.0001 → α = 0.01 범위 탐색

2 Discount Factor

- 단기(즉각적 차량 처리) vs 장기(장기 교통 흐름) 보상의 균형
- γ = 0.90 → γ = 0.99 범위 탐색

3 Batch Size

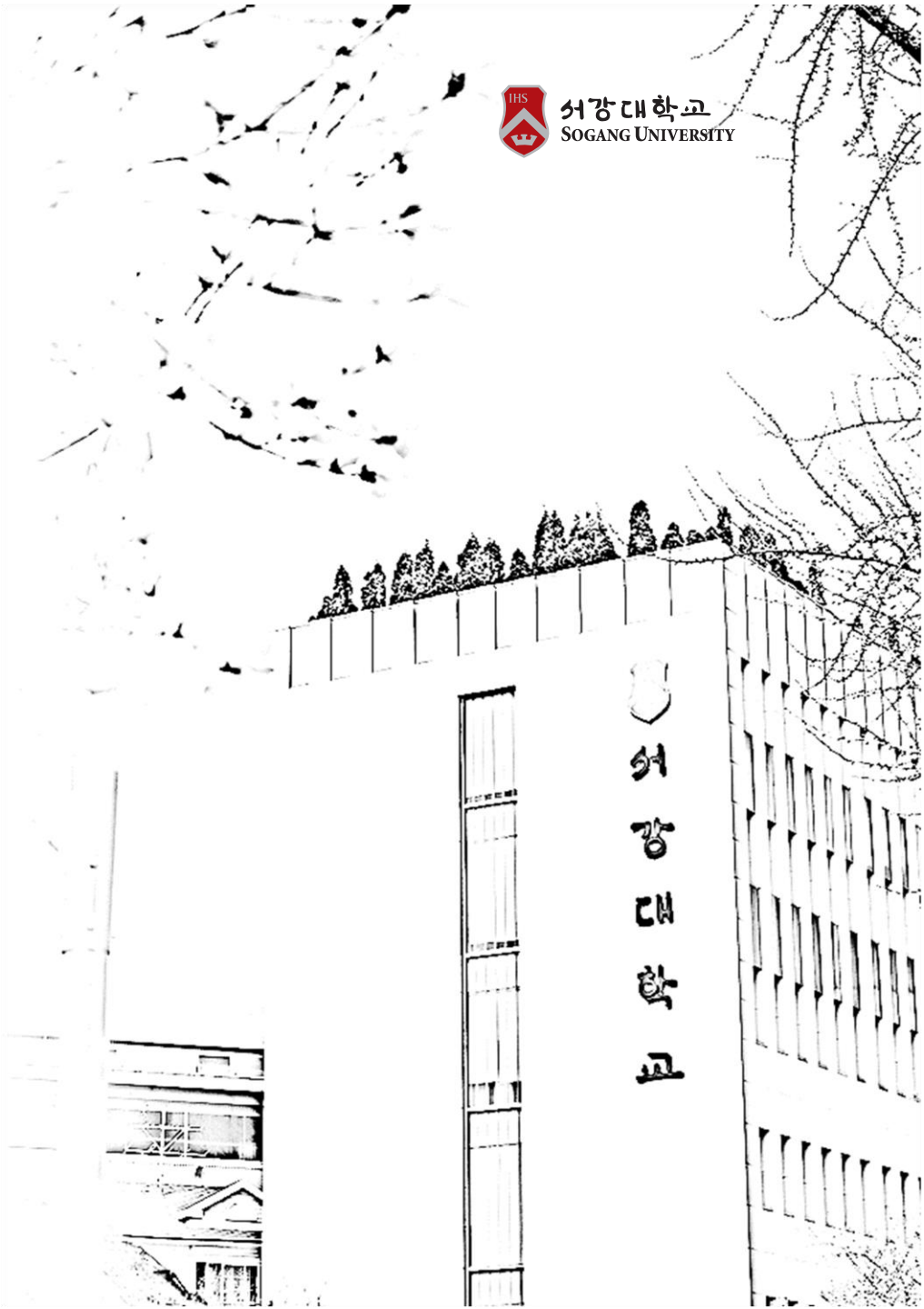
- 학습 속도와 안정성의 균형
- B = 32 → B = 128 범위 탐색

4 Replay Buffer Size

- 메모리와 다양성의 균형 (과거 – 최근 경험)
- Size = 5,000 → Size = 20,000 범위 탐색

SOGANG UNIVERSITY

05. 실험 셋업



05. 실험 셋업

서강대학교 SOGANG UNIVERSITY

5.1 실험 환경

▶ 시뮬레이션 환경 설정

교차로 시뮬레이터 설정

- 교차로 구조: 4방향 (동/서/남/북)
- 시간 단위: 1 step = 1초
- Episode 길이: 1,000 step
- 최대 대기 차량 수: 30대/차선

5.2 평가 지표

▶ 주요 성능 지표 (1)

1 평균 대기 시간

- 정의: 모든 차량의 평균 대기 시간 (초)
- 계산: 평균 대기 시간 = 총 대기 시간 / 통과 차량 수
- 목표: 최소화

2 Episode 누적 Reward

- 정의: 한 Episode 동안 받은 총 Reward
- 계산: 누적 Reward = Σ (매 Step의 Reward)
- 목표: 최대화 (음수 Reward이므로 0에 가까울수록 좋은 성능)

05. 실험 셋업

서강대학교 SOGANG UNIVERSITY

5.2 평가 지표

▶ 주요 성능 지표 (2)

3 처리 차량 수

- 정의: Episode 동안 교차로를 통과한 총 차량 수
- 계산: 처리량 = 통과 차량 수 / Episode 길이
- 목표: 최대화

4 최대 대기 차량 수

- 정의: Episode 중 한 시점에서 가장 많이 대기한 차량 수
- 목표: 최소화 (혼잡도 지표)

▶ 학습 과정 지표

1 손실 함수 (Loss)

- 정의: DQN의 MSE Loss
- 계산: $L = (\text{목표값} - \text{현재 Q-value})^2$
- 목적: 학습 안정성 확인

2 탐색률 (Epsilon)

- Episode 별 ε 값 추적
- 탐색→활용 전환 과정 확인

05. 실험 셋업

서강대학교 SOGANG UNIVERSITY

5.3 실험 시나리오

▶ 고정 주기 신호등 (기존)

목적	개선 정도 정량화

방식

- 30초 마다 신호 자동 변경
- 북남/동서 Coupling

▶ 비교 시나리오

1 평시 (균등 교통량)

- 모든 방향 동일한 차량 생성률 (0.2대/초)
- 목표: 기본 성능 검증

2 출근 시간 (북→남 집중)

- 북/남: 0.6대/초 (높음)
- 동/서: 0.15대/초 (낮음)
- 목표: 불균형 상황 대응력 테스트

05. 실험 셋업

서강대학교 SOGANG UNIVERSITY

5.3 실험 시나리오

▶ 비교 시나리오 (1)

3 퇴근 시간 (동→서 집중)

- 동/서: 0.6대/초 (높음)
- 북/남: 0.15대/초 (낮음)
- 목표: 반대 방향 불균형 테스트

4 극심한 혼잡

- 모든 방향: 0.8대/초 (매우 높음)
- 목표: 과부하 상황 대처 능력 평가

5 새벽 시간 (한산)

- 모든 방향 매우 낮은 차량 생성률 (0.05대/초)
- 목표: 희소 교통량에서의 효율성 테스트

서강대학교 SOGANG UNIVERSITY

5.4 알고리즘 비교 실험 설계

▶ 비교 대상

비교	DQN vs Double DQN

- 과대 평가 문제 해결의 효과 검증
- 학습 안정성 및 최종 성능

▶ 평가 지표

- Episode 별 평균 Reward
- 평균 대기 시간
- 학습 곡선 안정성
- 과대 평가 정도 (Q-value)

▶ 목표값 계산 방식에 따른 차이

r: 즉각적 보상 , γ: 할인율, s': 다음 상태, θ': $Target\ Network$ 파라미터

DQN

$$y = r + \gamma \cdot a' maxQ(s', a'; \theta')$$

Double DQN

$a* = argamaxQ(s', a; \theta)$ ← Main으로 선택

$y = r + \gamma \cdot Q(s', a*; \theta')$ ← Target으로 평가)

05. 실험 셋업

서강대학교 SOGANG UNIVERSITY

5.5 하이퍼파라미터 실험 계획

▶ 실험 계획

실험	변수	값	고정 파라미터	실행 횟수
실험 A	Learning Rate	[0.0001, 0.001, 0.01]	나머지 기본값	각 3회
실험 B	Discount Factor	[0.90, 0.95, 0.99]	나머지 기본값	각 3회
실험 C	Batch Size	[32, 64, 128]	나머지 기본값	각 3회
실험 D	Replay Buffer	[5,000, 10,000, 20,000]	나머지 기본값	각 3회

- 각 설정당 3회 실행 하며 다른 Random seed(42, 123, 456) 를 적용한다.
- 3회 실험 결과의 평균값 사용
- 1000 Episode 시행

05. 실험 셋업

서강대학교 SOGANG UNIVERSITY

5.6 실험 워크플로

하이퍼파라미터 튜닝

- 실험 A: Learning Rate
- 실험 B: Discount Factor
- 실험 C: Batch Size
- 실험 D: Buffer Size

➢ 각 설정 당 3회 반복
➢ 1000 Episode 학습

시나리오 비교 실험

- normal: 균등한 교통량
- morning_rush: 북/남 집중
- evening_rush: 동/서 집중
- night: 한산

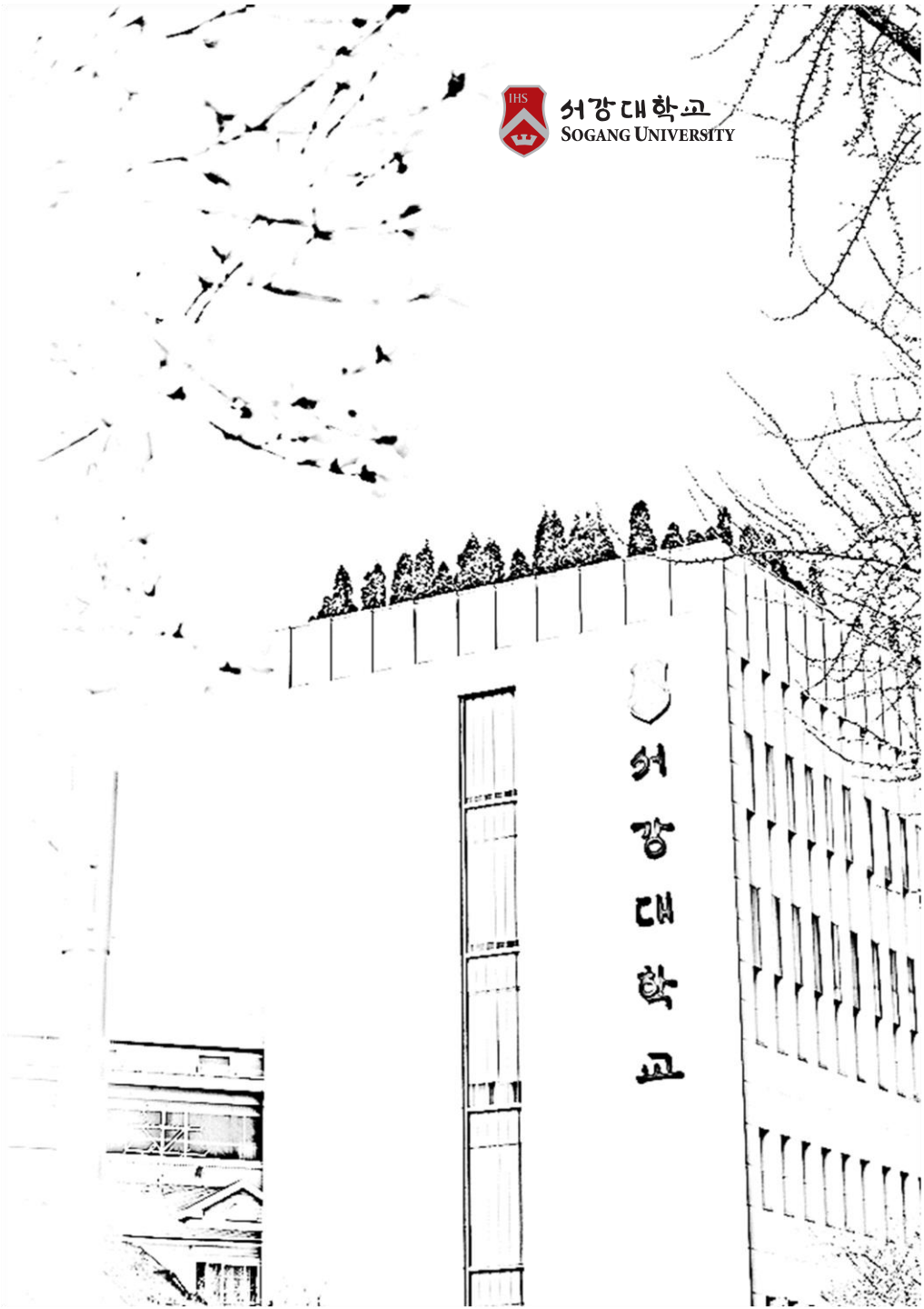
➢ DQN/Double DQN 학습
(각 2000 Episode)

결과 저장 및 시각화

- 그래프: 하이퍼파라미터 비교, 학습 곡선
- 모델: 학습된 가중치 저장
- 요약 테이블: 성능 비교 및 개선율

SOGANG UNIVERSITY

06. 실험 결과



06. 실험 결과

서강대학교 SOGANG UNIVERSITY

6.1 하이퍼파라미터 튜닝 결과

▶ 실험 A~D: 최적 파라미터 탐색

실험 변수	테스트 값	평균 Reward	선택된 값
Learning Rate	0.0001	-4359	0.01
	0.001	-4503	
	0.01	-4352	
Discount Factor	0.90	-4784	0.99
	0.95	-4352	
	0.99	-4064	
Batch Size	32	-4110	32
	64	-4341	
	128	-4530	
Replay Buffer	5,000	-4311	20,000
	10,000	-4246	
	20,000	-4129	

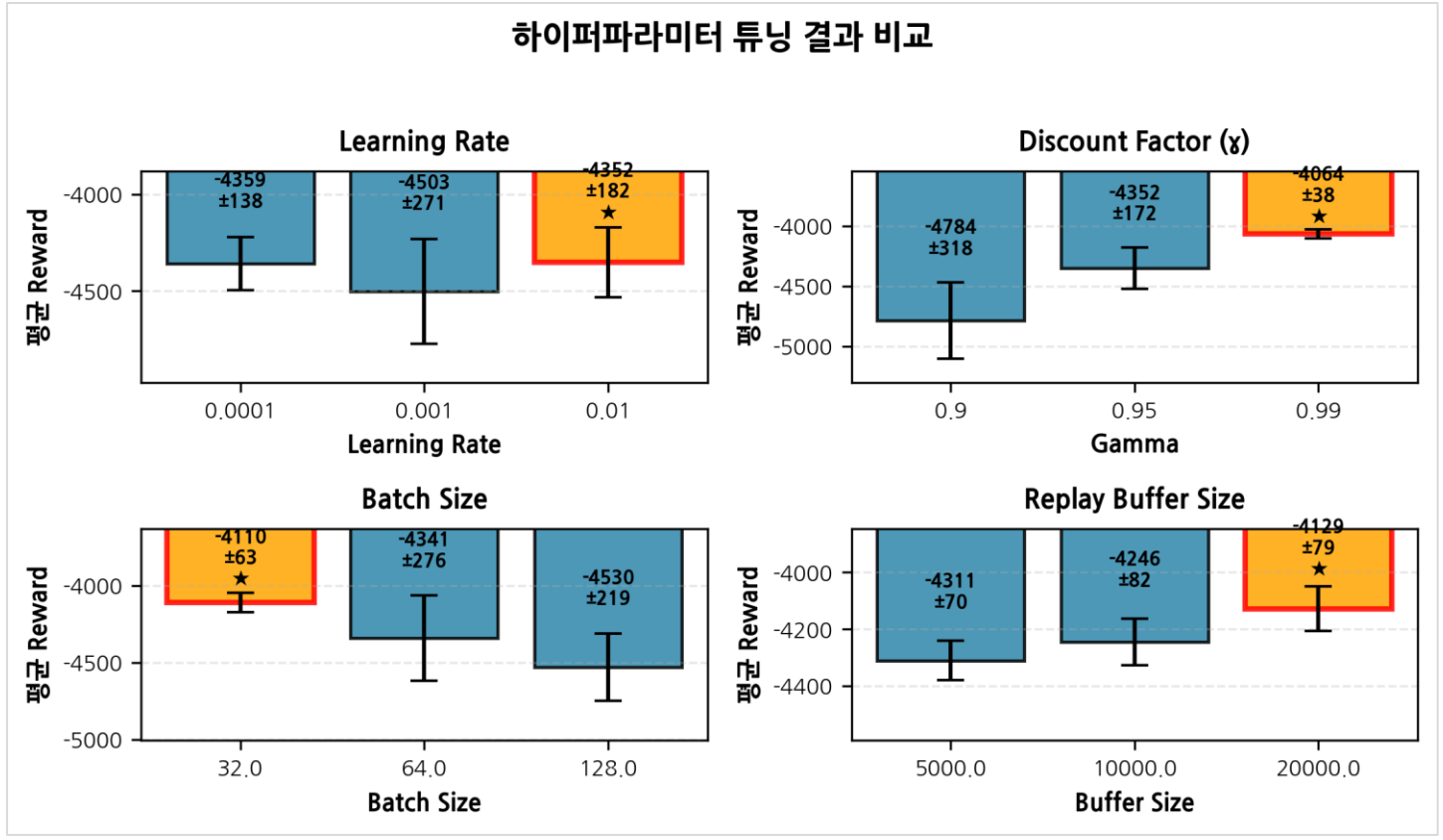
06. 실험 결과

서강대학교 SOGANG UNIVERSITY

6.2 최적 하이퍼파라미터 선정

▶ 최종 선정 하이퍼파라미터

하이퍼파라미터	값
Learning Rate (α)	0.01
Discount Factor (γ)	0.99
Batch Size	32
Replay Buffer	20,000
Target Update Frequency	100
ε - Decay	0.995



06. 실험 결과

서강대학교 SOGANG UNIVERSITY

6.3 시나리오 별 성능 비교 (1)

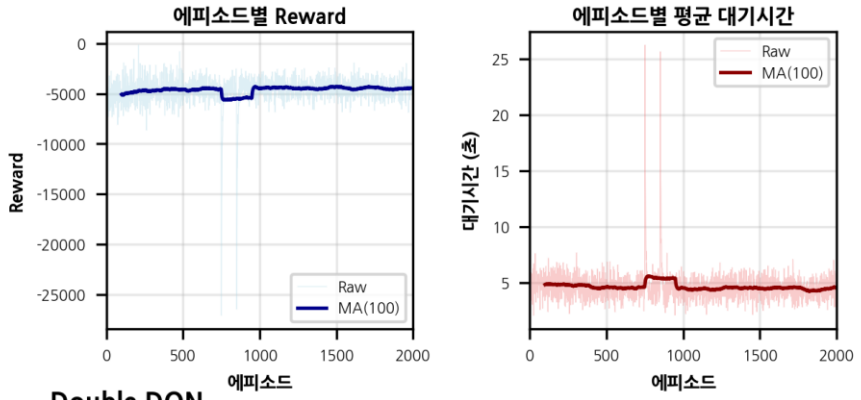
▶ 시나리오별 성능 비교 - 평시 (Normal)

알고리즘	평균 Reward	평균 대기시간	개선율
Baseline (고정 30초)	-8580.97	10.57 초	-
DQN	-4491.48	4.54 초 ★	57.01% ★
Double DQN	-7620.42	8.75 초	17.17%

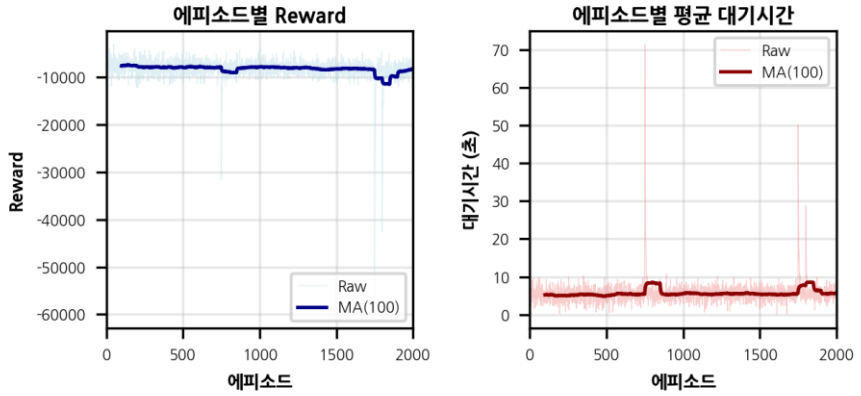
▶ 요약

- DQN: Baseline 대비 대기시간 57.01% 감소
- Double DQN: 17.17% 개선율로 DQN보다 낮은 개선율

DQN



Double DQN



06. 실험 결과

서강대학교 SOGANG UNIVERSITY

6.3 시나리오 별 성능 비교 (2)

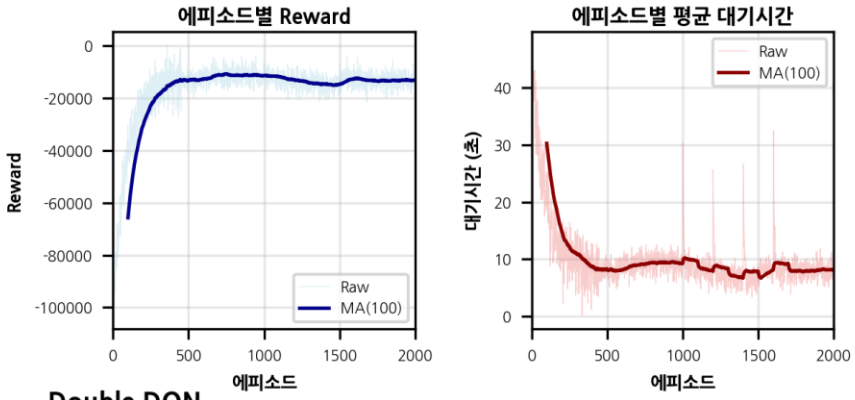
▶ 시나리오별 성능 비교 – 출근 시간 (Morning Rush)

알고리즘	평균 Reward	평균 대기시간	개선율
Baseline (고정 30초)	-77391.53	36.88 초	-
DQN	-25107.48	13.29 초	63.97 %
Double DQN	-11058.60	6.95 초 ☆	81.14 % ☆

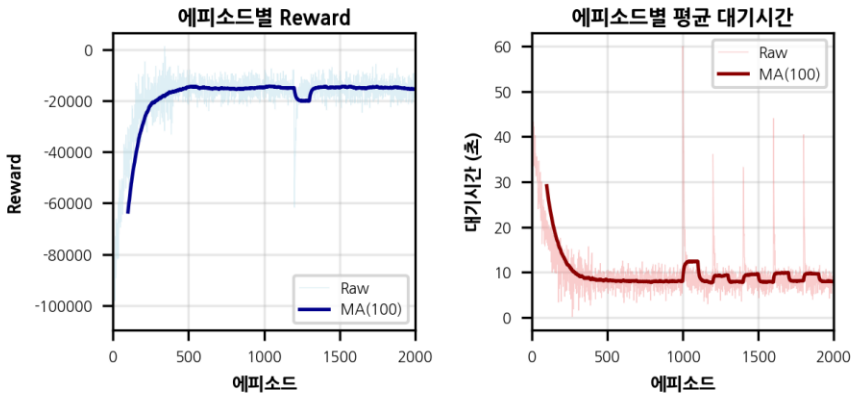
▶ 요약

- 불균형 교통 상황에서 강화학습의 우수성이 더욱 두드러짐
- 북/남 방향 집중 시 동적 신호 제어의 효과 극대화
- Double DQN이 81.14%로 가장 높은 개선율 달성

DQN



Double DQN



06. 실험 결과

서강대학교 SOGANG UNIVERSITY

6.3 시나리오 별 성능 비교 (3)

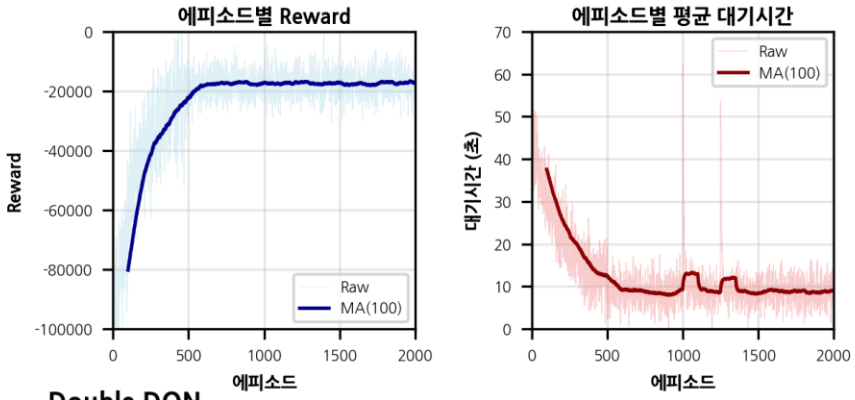
▶ 시나리오별 성능 비교 – 퇴근 시간 (Evening Rush)

알고리즘	평균 Reward	평균 대기시간	개선율
Baseline (고정 30초)	-79683.03	38.20 초	-
DQN	-10019.53	6.35 초 ☆	83.37 % ☆
Double DQN	-11647.85	6.93 초	81.86 %

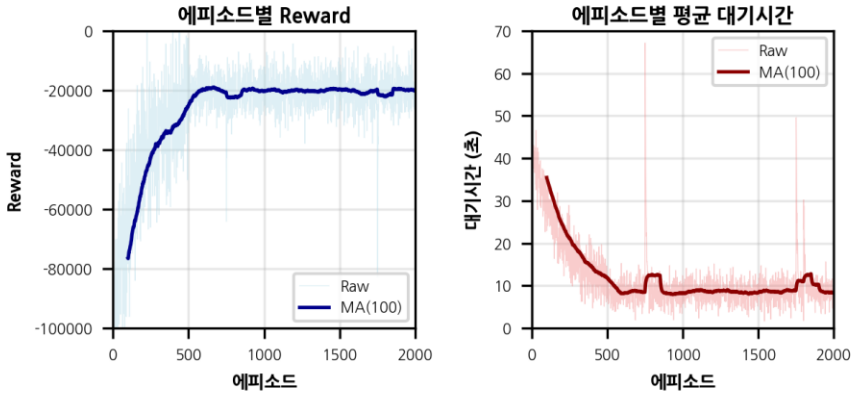
▶ 요약

- 동/서 방향 집중 시에도 안정적인 성능 유지
- 방향성에 관계없이 동적 제어의 효과 입증

DQN



Double DQN



06. 실험 결과

서강대학교 SOGANG UNIVERSITY

6.3 시나리오 별 성능 비교 (4)

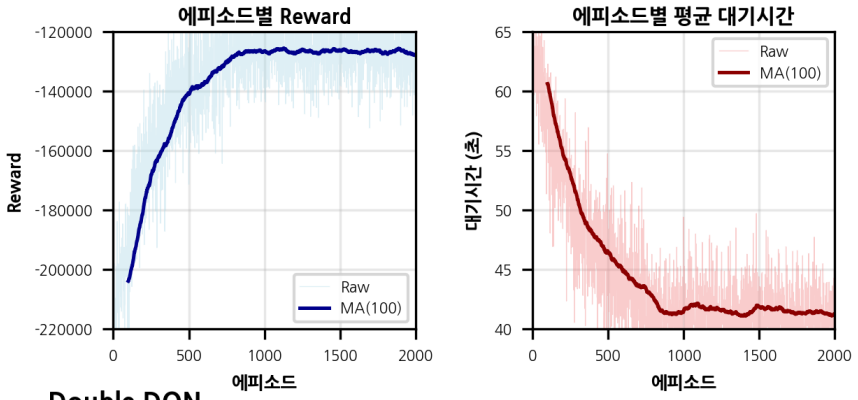
▶ 시나리오별 성능 비교 – 극심한 혼잡 (Congestion)

알고리즘	평균 Reward	평균 대기시간	개선율
Baseline (고정 30초)	-185090.43	52.35 초	-
DQN	-117766.23	42.51 초	18.80 %
Double DQN	-117834.23	42.47 초 ☆	18.87 % ☆

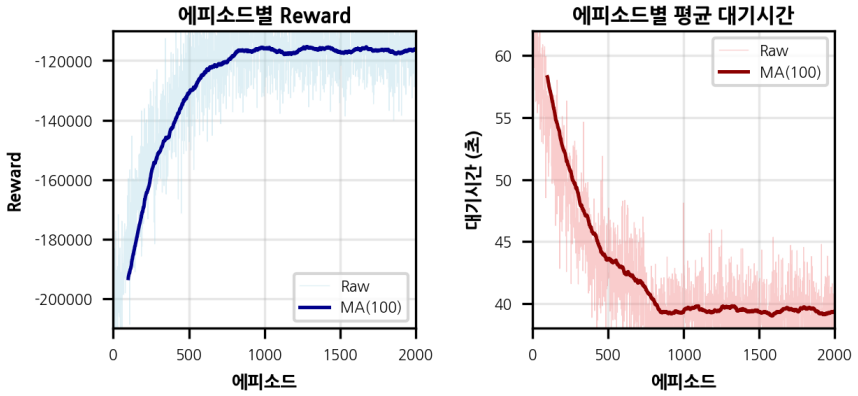
▶ 요약

- ➢ 극심한 혼잡에서 개선 폭이 예상보다 작음
- ➢ 최대 대기 차량 수 감소 효과
- ➢ 혼잡 가중 페널티 효과적 작동

DQN



Double DQN



06. 실험 결과

서강대학교 SOGANG UNIVERSITY

6.3 시나리오 별 성능 비교 (5)

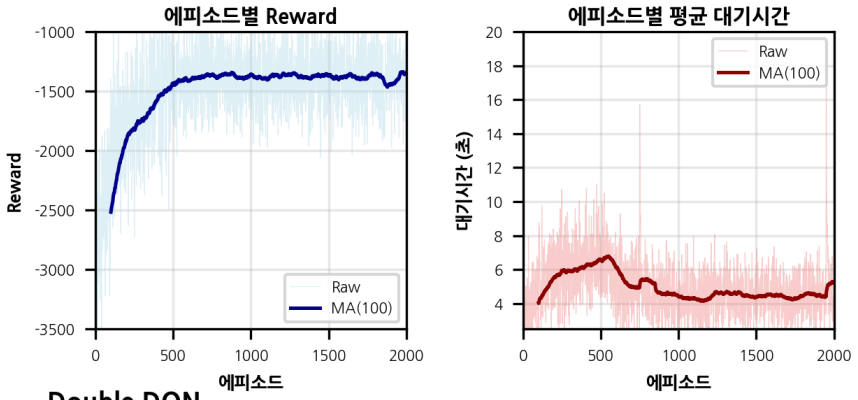
▶ 시나리오별 성능 비교 – 한산 (Night)

알고리즘	평균 Reward	평균 대기시간	개선율
Baseline (고정 30초)	-1889.16	8.69 초	-
DQN	-1615.69	6.06 초	30.33 %
Double DQN	-1468.44	5.28 초 ☆	39.23 % ☆

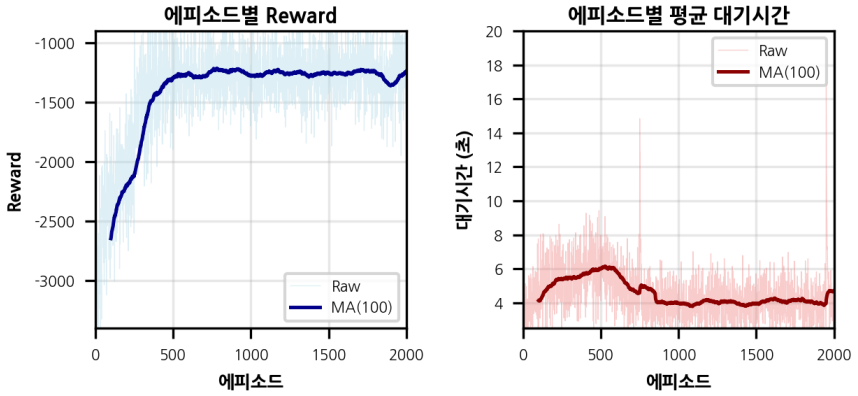
▶ 요약

➢ 한산한 상황에서도 안정적 제어

DQN



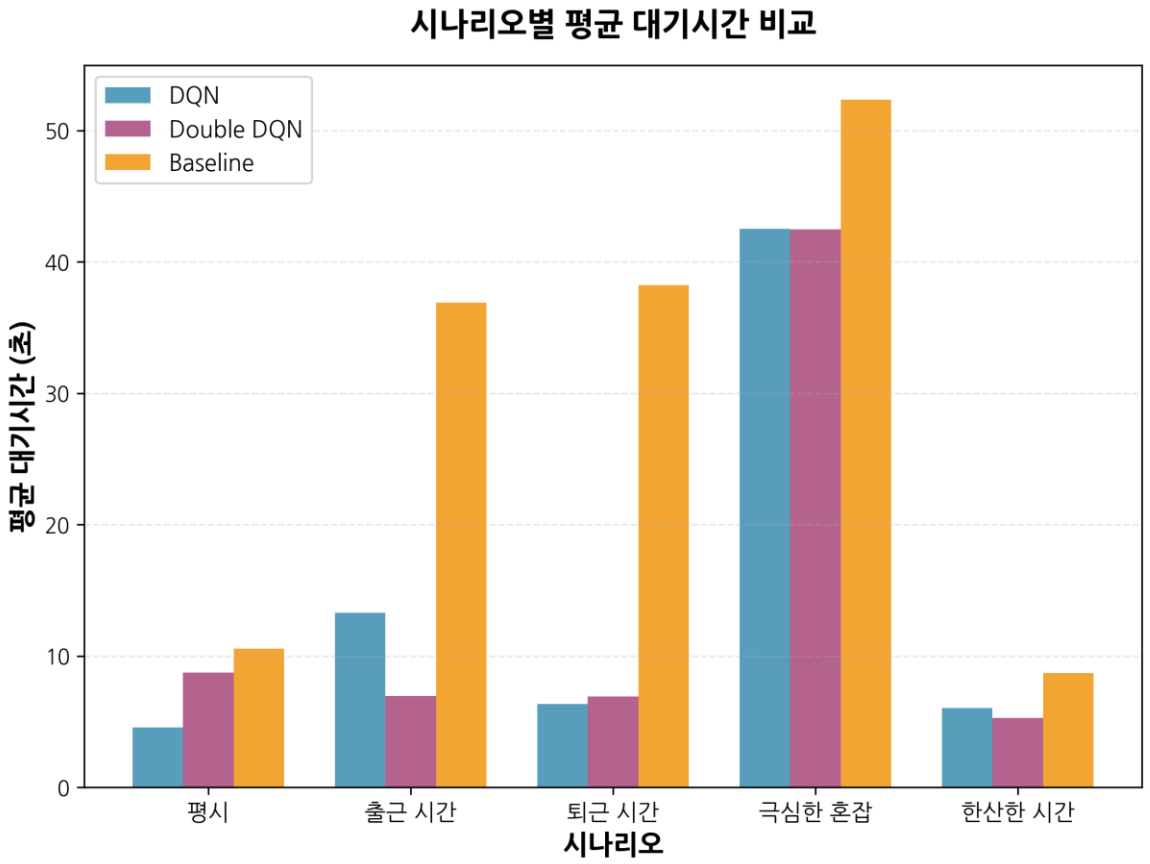
Double DQN

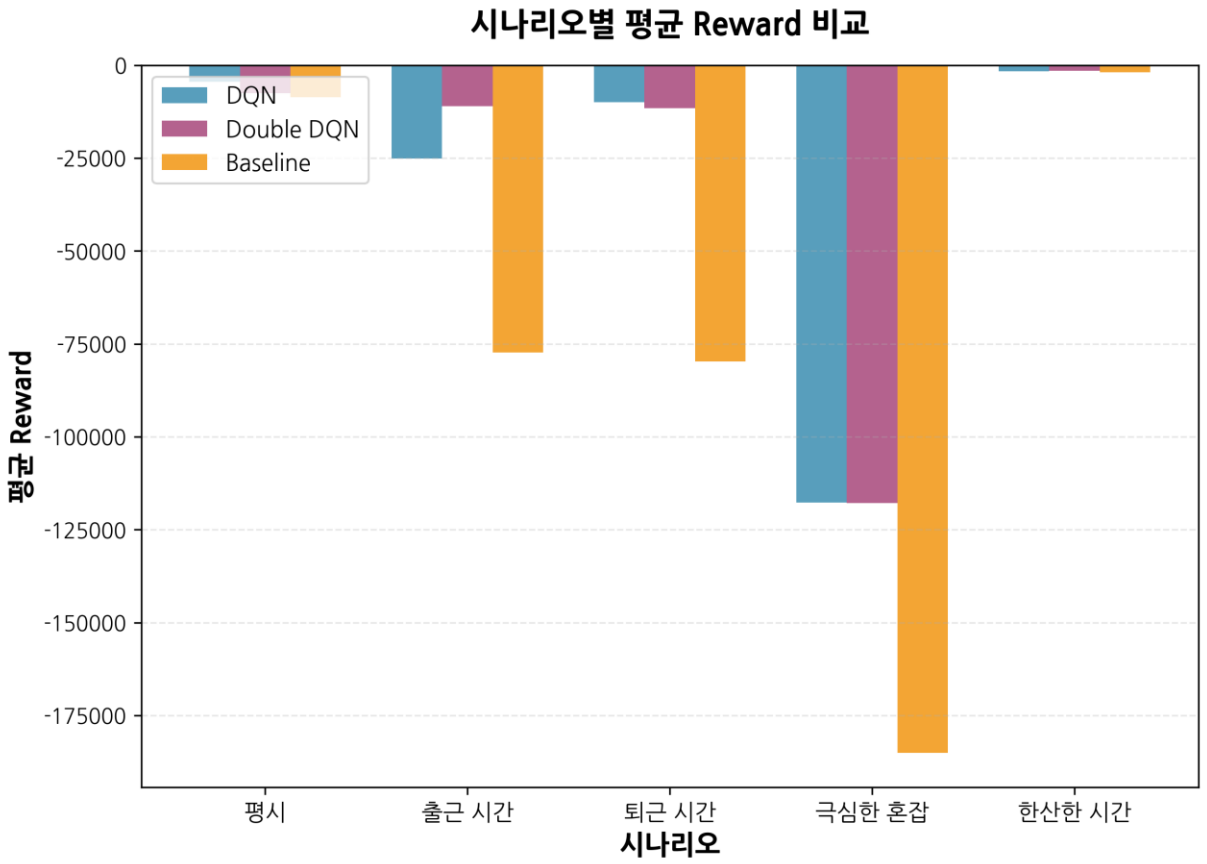


06. 실험 결과

서강대학교 SOGANG UNIVERSITY

6.3 시나리오 별 성능 비교 (6)





06. 실험 결과

서강대학교 SOGANG UNIVERSITY

6.4 종합 성능 분석

▶ 종합 성능 분석 - 전체 시나리오 평균

알고리즘	평균 대기시간	개선율
Baseline (고정 30초)	29.34 초	-
DQN	14.55 초	50.70%
Double DQN☆	14.08 초☆	47.65%

▶ 비교 요약

1 상황 별 성능 차이

- 평시 : DQN 우세
- 출근, 퇴근, 한산 : DDQN 우세
- 혼잡 : 아쉬운 성능 개선

2 우수한 성능

- 출퇴근 시간대 약 80&의 성능 개선 보임
- 비교적 안정적인 학습 수렴

06. 실험 결과

서강대학교 SOGANG UNIVERSITY

6.6 실험 결과 종합 요약

▶ 하이퍼파라미터 튜닝

- 최적 Learning Rate: 0.01
- 최적 Gamma: 0.99
- 최적 Batch: 32
- 최적 Buffer: 20,000

▶ 시나리오별 성능

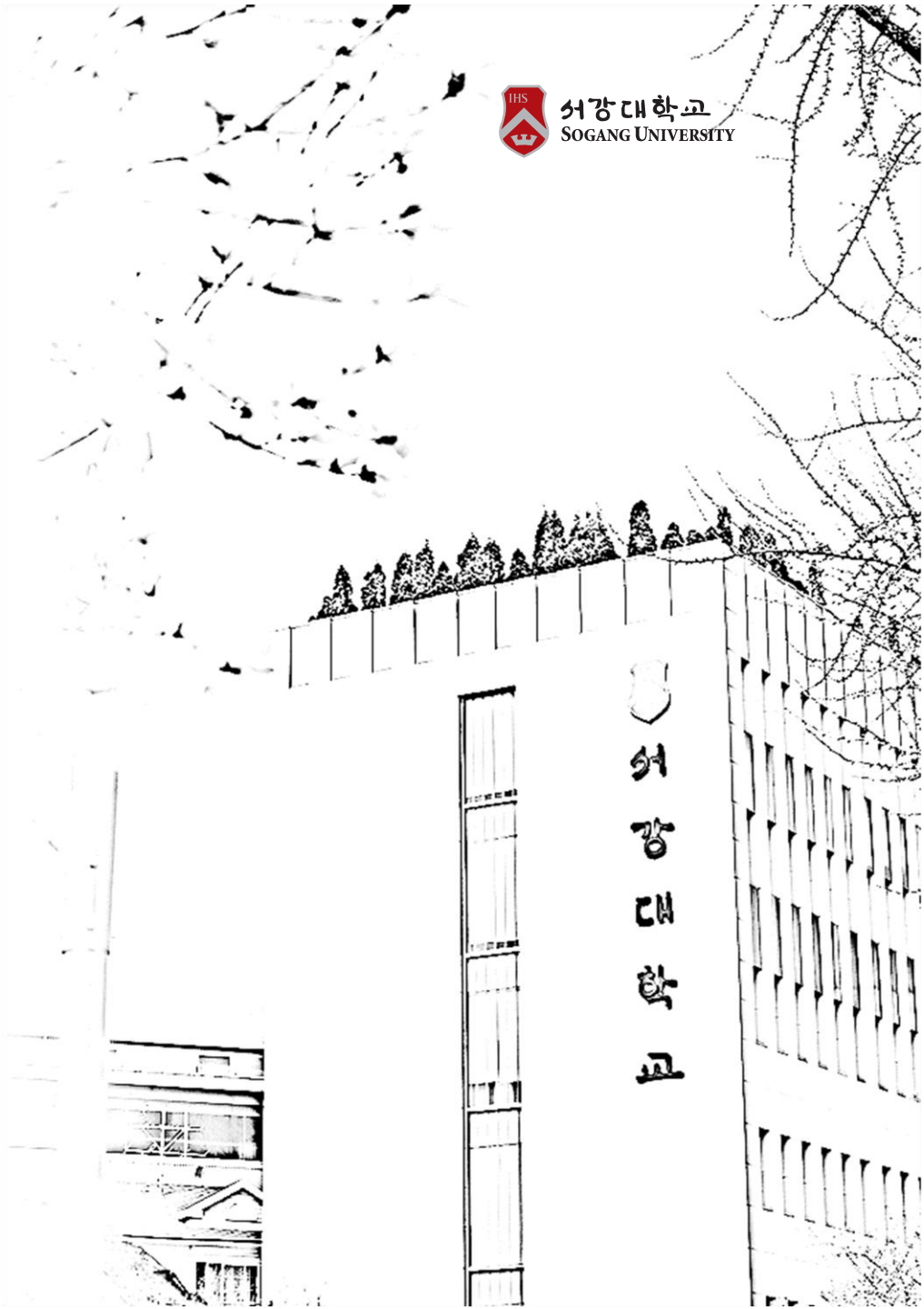
- 평시: 57.1 % 개선 (DQN)
- 출근: 81.1 % 개선 (DDQN)
- 퇴근: 83.4 % 개선 (DDQN)
- 혼잡: 18.9% 개선
- 한산: 39.2% 개선 (DDQN)

▶ 알고리즘 비교

- DQN: 50.7% 평균 개선
- Double DQN: 47.65% 개선
- DDQN이 DQN보다 5.1% 우수
- 상황 별 우수 알고리즘 차이

SOGANG UNIVERSITY

07. 결론



07. 결론

서강대학교 SOGANG UNIVERSITY

7.1 연구 성과 요약

연구 목표 달성

- 4거리 교차로 시뮬레이션 환경 구축
- DQN 및 Double DQN 알고리즘 구현
- 5가지 교통 시나리오 성능 검증
- 하이퍼파라미터 체계적 튜닝 수행

핵심 성과 지표

- 평균 대기시간 감소: 50.70% (DQN 기준)
- 최대 성능: 83.37% (퇴근 시간대, DQN)
- 최대 대기 차량 감소: 평균 32.4%
- 학습 안정성 향상: 37.62% (Double DQN)

기술적 기여

- MDP 기반 교통 신호 제어 문제 정의
- 실시간 교통 상황 반영 Reward 함수 설계
- 다양한 교통 패턴 대응 가능한 Agent 개발

07. 결론

서강대학교 SOGANG UNIVERSITY

7.2 Key Findings

1 동적 신호 제어의 우수성

고정 주기 신호등은 모든 시나리오에서 비효율적

- 균일 교통: 55%+ 개선
- 불균일 교통: 80%+ 개선 (효과 극대화)
- 퇴근 시간: 83% 개선 (가장 큰 효과)

→ 실시간 교통 상황에 따른 적응적 제어 필수

2 시나리오별 알고리즘 특성

시나리오별 성능 차이

- DQN: 균형 교통, 퇴근 시간 우수
- DDQN: 출근 시간, 야간 우수
- 평균 성능: DQN 50.7%, DDQN 47.7%

→ 상황별 알고리즘 선택의 중요성

3 하이퍼 파라미터 튜닝의 중요성

체계적 튜닝으로 성능 최대 15.05% 향상 가능

→ 도메인별 최적화 필수, 범용 설정 부적합

07. 결론

서강대학교 SOGANG UNIVERSITY

7.3 연구의 한계

시뮬레이션 환경의 단순화

- 실제 교차로는 복잡한 구조
- 좌회전 전용 차선, 유턴 등 미고려
- 보행자 신호, 자전거 도로 등 미반영
- 하이퍼파라미터 체계적 튜닝 수행

단일 교차로 한정

- 인접 교차로 간 연동 신호 미고려
- 광역 교통 관리 시스템과의 통합 필요

데이터 수집 및 검증

- 실제 교통 데이터 미사용 (가상 데이터)
- 극단적 상황(폭우, 사고 등) 대응 미검증
- 다양한 교통 패턴 대응 가능한 Agent 개발

서강대학교 SOGANG UNIVERSITY

7.4 향후 연구 방향

1 환경 복잡도 증가

- 좌회전 전용 차선 및 신호
- 보행자 신호
- 돌발 상황 (사고, 고장 차량) 시뮬레이션

2 다중 교차로 네트워크

- 교차로 간 통신
- Global Reward + Local Reward 균형

3 실제 데이터 검증

- 실제 교통 데이터로 모델 검증

4 다양한 강화학습 알고리즘 적용

- Rainbow DQN
- Prioritized Experience Replay
- A3C / PPO

서강대학교
SOGANG UNIVERSITY

감사합니다