Journal of KIIT. Vol. 24, No. 7, pp. 171-183, Jul. 31, 2026. pISSN 1598-8619, eISSN 2093-7571 
http://dx.doi.org/10.14801/jkiit.2026.24.7.171

# 심층강화학습 기반 저궤도 위성 네트워크 라우팅을 위한 CNN과 GNN의 성능 비교

김범석*, 노봉수**[1], 한명훈**[2], 이헌철***, 김성렬****

# Performance Comparison of CNN and GNN for Deep Reinforcement Learning-based Routing in LEO Satellite Networks

Bumseok Kim*, Bongsoo Roh**[1], Myoung-Hun Han**[2], Heoncheol Lee***, and Sungryul Kim****

이 논문은 2024 년 정부(방위사업청)의 재원으로 국방과학연구소의 지원을 받아 수행된연구임(UI247034TF)


## 요 약

저궤도(LEO) 위성 네트워크 라우팅은 빠른 토폴로지 변화와 다중 홉 부하 분산, 실시간 의사결정 제약으로 지상망보다 복잡하다. 심층강화학습 기반 라우팅에서는 상태 표현 방식이 핵심 설계 요소이며, 위성 토폴로지가 그래프 구조를 가지므로 기존 연구 대부분은 그래프 신경망을 채택해왔다. 그러나 추론 속도와 임베디드 하드웨어 관점에서는 고정 입력 형태를 갖는 합성곱 신경망이 유리할 수 있다. 본 연구는 DQN(Deep Q-Network) 환경에서 두 표현 모델을 적용하여 성능을 비교하였다. GNN의 처리량이 CNN보다 15.2% 높았으나, CNN의 추론 지연은 2.84 ms로 GNN 대비 약 2.3배 빨랐다. 본 연구는 해당 실험 조건에서 CNN과 GNN 적용을 위한 설계 지침을 제공하며, 두 표현이 라우팅 처리량과 추론 속도 간 상충관계를 가짐을 확인하였다.

## Abstract

Routing in Low-Earth-Orbit (LEO) satellite networks is more complex than terrestrial routing due to rapid topology changes, multi-hop load balancing, and real-time decision constraints. State representation is crucial for DRL-based routing, and the graph-structured nature of satellite topology has led most prior studies to adopt GNNs.. From the perspective of inference speed and embedded deployment, however, CNNs with fixed-size inputs can be more favorable.. We compare the two representations in an identical DQN environment. While GNNs achieved 15.2% higher throughput than CNNs, CNNs achieved an inference latency of 2.84 ms, which was approximately 56.5% lower than that of GNNs. This study provides design guidelines for applying CNNs and GNNs under the studied experimental conditions and confirms a trade-off between routing throughput and inference speed.



Keywords

LEO satellite network, deep reinforcement learning, routing, convolutional neural network, graph neural network



* 국립금오공과대학교 소프트웨어공학과 석사과정
- ORCID: https://orcid.org/0009-0001-5634-3227
** 국방과학연구소 선임연구원
- ORCID[1]: https://orcid.org/0009-0001-9517-8885
- ORCID[2]: https://orcid.org/0009-0004-2571-0153
*** 국립금오공과대학교 IT융복합공학과 교수
- ORCID: https://orcid.org/0000-0003-2962-3474
**** 국립금오공과대학교 컴퓨터공학부 부교수(교신저자)
- ORCID: https://orcid.org/0009-0005-5218-1477



· Received: Jun. 10, 2026, Revised: Jul. 13, 2026, Accepted: Jul. 16, 2026


· Corresponding Author: Sungryul Kim
Dept. of Software, School of Electronic Engineering Kumoh National Institute of Technology, Korea
Tel.: +82-54-478-7549, Email: sungryul@kumoh.ac.kr

## Ⅰ. 서 론

저궤도(LEO, Low-Earth-Orbit) 위성 통신은 SpaceX Starlink와 Amazon Kuiper가 주도하는 군집 위성 네트워크의 등장으로 전례 없는 규모로 확장되고 있다. 대규모 LEO 망은 지상망의 음영 지역 해소, 광역 모바일 백홀, 항공, 해상 통신 등 다양한 응용에서 핵심 인프라로 자리매김하고 있으며, 다중 홉 위성 간 링크(ISL, Inter-Satellite Link)를 통해 광역 라우팅 인프라를 형성한다[1].

위성 라우팅은 지상망 라우팅과 본질적으로 다른 세 가지 제약이 있다. 첫째, LEO 위성은 시속 약 27,000 km로 공전하기 때문에 토폴로지가 수십 초 단위로 변한다[2]. 둘째, 광역 통신을 위해 다중 홉을 거치므로 단일 경로 최소화는 부하 분산을 깨뜨려 핫스팟을 유발한다[3]. 셋째, 실시간 의사결정이 필요하여 복잡한 최적화 기법을 적용할 여유가 없다. 전통적인 Dijkstra 계열 최단 경로 알고리즘과 그 변형은 정적인 가중치에 의존하므로 동적 부하 변화에 대응하지 못한다[1].

이러한 제약을 극복하기 위해 최근 심층 강화학습(DRL, Deep Reinforcement Learning) 기반 라우팅이 활발히 연구되고 있다. 대표적으로 그래프 신경망(GNN, Graph Neural Network)과 DRL을 결합한 GRouting[4], GraphSAGE 기반 인덕티브 라우팅 정책[5], 분산 다중 에이전트와 지속 학습을 결합한 연속 DRL 라우팅[6], 초고밀도 LEO 망을 위한 다중경로 협력 라우팅[7] 등이 제안되었다. DRL 기반 라우팅에서는 네트워크 상태를 정책의 입력으로 변환하기 위해 CNN 또는 GNN과 같은 표현 계층을 사용하며, 네트워크 토폴로지가 노드와 엣지의 연결로 자연스럽게 표현된다는 특성으로 인해 대부분의 연구가 GNN을 채택하고 있다.

그러나 GNN 표현은 두 가지 실용적 제약을 갖는다. 첫째, 군집 규모가 가변적인 환경에서는 입력 텐서의 모양이 노드와 엣지 수에 비례해 변하므로 임베디드 가속기에서 정적 컴파일을 위한 최적화가 제한될 수 있다.

둘째, 메시지 패싱(Message passing) 연산이 매 계층마다 노드 수 규모의 인접 행렬 연산을 반복하여 연산량이 누적되어, 결과적으로 합성곱 신경망(CNN, Convolutional Neural Network) 기반 모델보다 추론 지연이 높아질 수 있다. 저궤도 위성 네트워크는 레이저 기반 위성 간 링크(ISL, Inter-Satellite Link)를 통해 대용량 데이터를 고속으로 전송할 수 있으므로, 라우팅 의사결정의 추론 속도 또한 중요한 성능 지표가 된다[8].

CNN은 입력이 고정 크기 텐서로 표현되어 정적 토폴로지 컴파일이 용이하고, 규칙적 메모리 접근으로 FPGA 가속에 구조적으로 유리할 수 있다. 다만 격자 형태의 고정 입력 구조는 학습 시와 다른 토폴로지가 주어질 경우 모델 재학습 또는 구조 변경이 요구되며, 비격자 연결 관계를 격자 좌표에 사상하는 과정에서 일부 위상 정보가 유실될 수 있다는 한계도 존재한다. 이처럼 일반 네트워크 라우팅에서는 GNN 표현이 CNN과 같은 유클리드 기반 표현보다 미관측 토폴로지에 더 잘 일반화됨을 보인 비교 연구가 보고된 바 있으나[9], 저궤도 위성 라우팅 환경에서 동일 조건으로 CNN과 GNN 표현을 직접 비교한 사례는 아직 많지 않다.

이에 본 연구에서는 사전 계산 경로를 액션 공간으로 하는 DQN(Deep Q-Network) 환경에서 CNN과 GNN의 상태 표현 처리량 및 추론 속도를 비교한다. 144개 위성 규모의 실제 궤도 기반 LEO 토폴로지에 부하 집중 시나리오와 무작위 링크 장애 모델을 적용하여 균일 트래픽 가정을 넘어 실제 망에 근접한 환경을 구성하고, 두 표현을 처리량, 추론 지연, 모델 파라미터 수의 세 축에서 비교한다. 이를 통해 저궤도 위성 네트워크 환경에서 심층강화학습 기반 라우팅에 적용되는 CNN과 GNN의 성능 및 추론 효율을 분석하고, 상태 표현 모델 선택을 위한 설계 지침을 제공한다.

본 논문의 구성은 다음과 같다. 2장에서 강화학습 기반 라우팅과 DRL 표현 학습 관련 연구를 정리한다. 3장에서 환경 설계와 CNN/GNN 모델 구조를 제시한다. 4장에서 성능 평가 결과를 분석하고, 5장에서 결론과 향후 과제를 기술한다.

## Ⅱ. 관련 연구

### 2.1 강화학습 기반 라우팅

Boyan과 Littman[10]은 1993년 Q-learning을 패킷 라우팅에 적용한 Q-routing을 제안하여 강화학습 기반 라우팅의 시초를 열었다. 이후 Mnih er al.[11]이 제안한 DQN이 강화학습에 심층 신경망을 도입하였고, 이를 기반으로 Double DQN[12], Dueling Network, 우선순위 경험 재생(PER, Prioritized Experience Replay) 등 다양한 개선 기법이 결합되어 안정성과 표본 효율이 크게 향상되었다[13][14]. 이 흐름은 위성 망에도 확산되어, DQN 기반 부하 분산을 시작으로 GNN 표현 학습이 표준 패러다임이 되었다[15]. 한편 위성 라우팅에서는 토폴로지의 시변성을 다루기 위해 스냅숏 기반 최단 경로, 가상 토폴로지, 부하 분산 휴리스틱 등 규칙 기반 기법이 제안되어 왔으며, 최근에는 이러한 기법의 한계를 보완하기 위해 학습 기반 접근이 도입되고 있다.

### 2.2 표현 학습: GNN

GNN은 노드와 엣지로 구성된 그래프를 입력으로 받아, 각 노드가 이웃 노드의 특징을 반복적으로 취합(Aggregation)하며 자신의 임베딩을 갱신하는 메시지 패싱(Message passing) 구조를 따른다[16]. 위성 네트워크에 적용할 때는 각 위성을 노드로, 위성 간 링크(ISL)를 엣지로 두고, 노드 특징으로 큐 길이, 위치, 잔여 용량 등을, 엣지 특징으로 링크 지연, 가용 대역폭 등을 부여하여 그래프를 구성한다.

토폴로지가 시간에 따라 변하더라도 인접 행렬만 갱신하면 동일한 모델을 재사용할 수 있어, 노드 수와 연결 구조가 달라지는 LEO 환경에 부합한다. 이러한 특성으로 인해 위성 라우팅 연구 대부분은 GNN 표현을 채택해왔다[17].

GNN의 표현력은 메시지 패싱의 반복 횟수, 즉 홉(Hop) 수에 좌우된다. K회 반복하면 각 노드는 K-홉 이내 이웃의 정보를 임베딩에 반영하므로, 홉 수를 늘릴수록 더 멀리 있는 위성의 상태까지 포착하여 전역 혼잡과 우회 경로를 고려한 라우팅이 가능하다. 그러나 홉 수가 과도하면 노드 임베딩이 서로 유사해져 노드 고유 정보가 희석되는 과도한 평활화(Over-smoothing)가 발생하고, 반복 연산이 누적되어 추론 지연이 증가한다[18]. 따라서 위성 라우팅에서 홉 수는 원거리 정보의 포착 범위와 정보 유실, 연산 비용 간의 절충을 고려해 결정해야 하는 핵심 설계 변수이다.

### 2.3 표현 학습: CNN

CNN은 입력이 고정 크기 격자 텐서로 주어진다는 점에서 정적 컴파일과 규칙적 메모리 접근에 유리하나, 본질적으로 그래프인 위성 토폴로지를 격자에 표현하는 설계가 선행되어야 한다. 사상 방식에 따라 표현 가능한 정보와 효율이 달라지므로, 격자 구성은 CNN 기반 상태 표현에서 가장 중요한 설계 결정이다. 본 절에서는 두 가지 방식을 검토한다.

첫째는 위성 수 × 위성 수 격자 방식이다. N개의 위성에 대해 N×N 격자를 구성하고, (i, j) 원소에 위성 i와 j 사이의 연결 여부, 링크 지연, 트래픽량 등의 관계를 채운다. 그래프를 고정 크기 격자로 변환해 표준 합성곱을 적용하는 이 접근은 인접 행렬을 그대로 이미지처럼 다루므로 위성 간 쌍(pair) 관계를 직접 표현하지만[19], 그림 1과 같이 격자 크기가 위성 수의 제곱으로 증가하여 규모가 커질수록 입력 차원과 연산량이 빠르게 늘어난다.

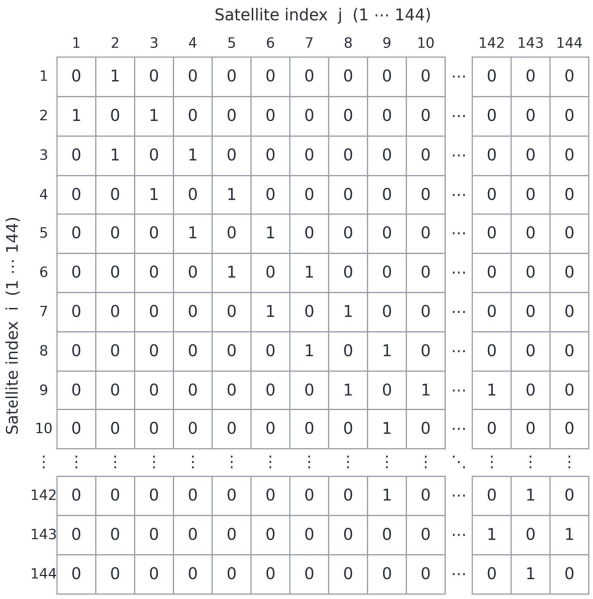



그림 1. 144×144 인접행렬 기반 토폴로지 표현
Fig. 1. Adjacency-matrix representation (144×144)

둘째는 그림 2와 같이 궤도 구조를 기준으로 위성을 격자에 배치하는 방식이다. 각 위성이 속한 궤도면을 행으로, 해당 궤도 내에서의 순서를 열로 삼아 144개 위성을 12×12 격자에 매핑한다. 위성 간 링크가 궤도 내·궤도 간 연결로 구성되는 Grid 구조에서[3], 행 인덱스는 궤도면 번호에, 열 인덱스는 그 궤도 내 위성의 순서에 대응하므로 같은 궤도면의 위성끼리 한 행에 놓인다. 본 연구에서는 위성이 존재하지 않는 격자 위치를 가상 노드(Virtual node)로 채워 입력 격자의 크기를 일정하게 유지한다.

이 방식의 이점은 격자의 행렬 구조 자체가 위성망의 물리적 배치를 반영한다는 데 있다. 인접 행렬을 단순히 이미지로 다루는 첫째 방식이 위성 간 쌍관계만을 담는 것과 달리, 궤도면을 행으로 정렬하면 같은 궤도면 내 이웃 위성은 같은 행에서 인접하고 인접 궤도면의 위성은 이웃한 행에 놓이게 된다.

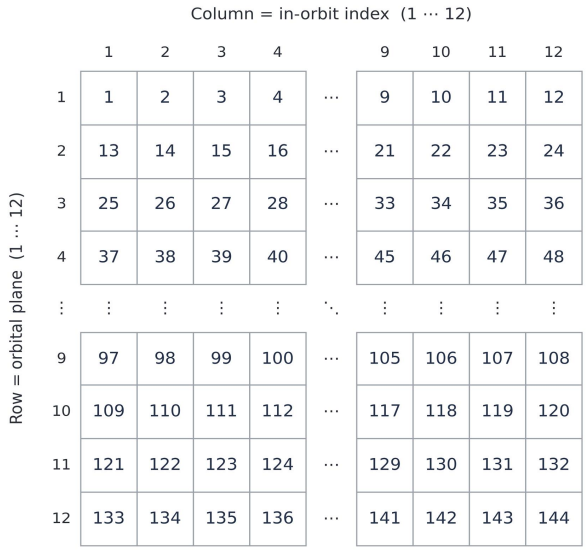



그림 2. 12×12 가상 노드 격자 표현
Fig. 2. Virtual-node grid representation (12×12)

따라서 합성곱의 공간 국소성이 궤도면 간 연결이라는 물리적 이웃 관계와 직접 대응하여, 격자 표현이 위성망의 위상 구조를 자연스럽게 담아낼 수 있다. 또한 144개 위성이 12개 궤도면 × 궤도면 당 12기로 구성되어 격자의 모든 칸이 실제 위성에 대응하므로 고정 크기 입력이 보장된다. 본 연구는 이러한 확장성과 위상 정합성을 고려하여 12×12 궤도 격자 표현을 채택하였다.

## III. 제안 방법

### 3.1 DQN 구조

본 연구는 저궤도 위성 네트워크의 동적 토폴로지에서 트래픽 요청을 라우팅하는 문제를 마르코프 결정 과정(MDP, Markov Decision Process)으로 모델링하고, DQN(Deep Q-Network)을 통해 라우팅 정책을 학습한다. 매 요청마다 에이전트는 사전 계산된 K개의 후보 경로 중 하나를 선택하며, 해당 경로를 따라 위성 간 링크(ISL)의 잔여 용량을 점유한다.

MDP는 다음과 같이 정의된다. 상태 S는 현재 토폴로지, 링크 잔여 용량, 트래픽 요청의 출발지와 목적지 정보, 후보 경로 정보를 포함한다. 행동 A는 K=4개의 후보 경로 중 하나를 선택하는 것으로, A = {0, 1, 2, 3}이다. 보상 R은 전달 성공 시 잔여 용량, 지연, 홉 수, 경로 중복을, 실패 시 장애, 병목 페널티를 반영한 스칼라 값이다.

상태 전이는 선택된 경로의 모든 링크에서 요청 수요만큼 용량을 차감한 뒤 다음 요청으로 진행한다. 상태 표현과 보상의 구체적 정의는 각각 3.2절, 3.3절과 3.1절에서 상세히 기술한다.

학습 안정성을 확보하기 위해 본 연구에서는 Double DQN[12], Dueling Network[13], 우선순위 경험 재생(PER)을 결합한 표준 구조를 채택하였다[14]. Double DQN은 가치 함수의 과대 추정을 완화하고, Dueling 구조는 상태 가치 V(s)와 행동 이점 A(s, a)를 분리하여 학습 효율을 높이며, PER은 학습 가치가 높은 경험을 우선적으로 샘플링한다. 또한 타깃 네트워크는 소프트 타깃 업데이트(Soft target update)로 점진적으로 갱신한다.

#### 3.1.1 환경

본 연구에서 다루는 LEO 위성 네트워크는 12개의 궤도면과 각 궤도면당 12기의 위성으로 구성되어 총 144기의 위성을 포함하는 격자 토폴로지를 가정한다. 학습 환경에서 한 타임스텝은 10초에 해당하며, 전체 실험은 총 1,081개의 타임스텝으로 구성된다. 인접 위성 간 ISL은 동일 궤도면 내 남북

방향 2개와 인접 궤도면의 동서 방향 2개로 최대 4개를 가지며, 각 타임스텝마다 위성 위치(x, y, z)와 144×144 인접 행렬이 미리 주어진다.

이때 출발 위성은 한국 상공을 중심으로 반경 4,500 km 이내에 위치한 위성 중에서 선정하여 트래픽이 영역에 집중되는 부하 편중 상황을 모사하며, 목적지 위성은 해당 출발 위성과 실제 통신이 이루어질 수 있는 인접 위성 중에서 선정한다.

한 에피소드는 40개의 타임스텝 동안 진행되며, 매 타임스텝마다 10개의 출발-목적지 위성 쌍 요청이 발생한다. 따라서 각 ISL의 초기 용량을 500으로 할당하고, 각 요청은 150만큼의 고정된 자원 수요를 가지며, 선택된 경로상의 모든 위성 간 링크(ISL) 용량에서 해당 수요만큼 차감된다. 잔여 용량이 0 이하가 된 링크는 장애 링크로 표시되어, 같은 타임스텝의 이후 요청에서는 사용이 제한된다.

3.1.2 행동

라우팅 에이전트는 트래픽 요청 (src, dst, t)이 도착할 때 K=4개의 사전 계산 후보 경로 중 하나를 행동 공간 A = {0, 1, 2, 3}에서 선택한다. 단일 최단 경로만 사용하지 않고 다중 경로 후보를 두는 이유는, 동적 토폴로지에서 특정 경로가 혼잡하거나 단절되었을 때 부하를 다른 경로로 분산시키기 위함이다.

이때 후보 경로들이 링크나 노드를 과도하게 공유하면 다중 경로 구성의 의미가 약해진다. 공유 링크 하나가 실패할 경우 여러 후보가 동시에 차단되어 선택지가 사실상 하나로 축소되기 때문이다. 따라서 후보 경로 간 겹침을 최소화하여 부하 분산과 장애에 대한 강인성을 함께 확보해야 한다.

본 연구는 엣지 페널티 기반 반복 최단 경로 알고리즘으로 K개의 경로를 생성한다. 초기 가중치 1로 시작하여 출발지→목적지 최단 경로를 계산하고, 사용된 엣지에 페널티를 누적해 가중치를 갱신한 뒤 다음 경로를 탐색하는 절차를 K회 반복한다. 경로 내 순환은 사후 제거 단계에서 처리한다.

엣지 페널티는 노드 페널티 방식과 비교할 때 경로 길이의 균형 측면에서 유리하다. 노드 단위 페널티는 해당 노드 전체를 우회해야 하므로 경로 길이가 비대칭적으로 증가하는 반면, 엣지 단위 페널티는 사용된 링크만 회피하므로 다양성을 확보하면서도 K개 경로의 홉 수를 균형 있게 유지할 수 있다.

3.1.3 보상

보상 함수는 단순히 요청을 목적지에 전달하는 것을 넘어, 링크 장애와 용량 병목을 능동적으로 회피하면서 여유 용량이 있는 짧은 경로를 선택하도록 설계하였다. 요청 단위 보상은 전달 성공과 실패가 서로 배타적인 분기로 정의되며, 한 요청은 성공 시 식 (1)의 가중합 보상을, 실패 시 식 (2)의 단일 페널티를 받는다. 또한 에피소드 종료 시 식 (3)의 종료 보상을 1회 가산한다.

$$R_{succ} = 0.5 - 0.15d - 0.1h + 0.5\rho - 0.15o \tag{1}$$

$$R_{fail} = \begin{cases} -1.5(1+0.5p) & no\_capacity \\ -1.5 & fail\_link \\ -1.5 & otherwise \end{cases} \tag{2}$$

$$R_{term} = 0.05 \times N_{succ} \tag{3}$$

여기서 $\rho$는 경로에서 잔여 용량이 가장 작은 병목 링크의 잔여 용량 비율, $d$는 정규화 전파 지연, $h$는 정규화 홉 수, $o$는 K개 후보 경로 간 중간 노드의 중복 비율이며, 모두 [0, 1] 범위로 정규화된다. 식 (4) 처럼 $d$는 경로 전파 지연을 관측된 최대 지연으로, $h$는 홉 수를 최대 홉 수로, $o$는 후보 경로 간 공유 중간 노드 수를 경로 길이로 나누어 각각 정규화한다. 각 변수의 정규화 기준값은 표 1에 정리하였다. 성공 보상의 기본값은 0.5로, 그 값을 낮게 두어 전달 성공 자체보다 장애, 혼잡 회피가 더 지배적인 학습 신호가 되도록 하였다. 잔여 용량 $\rho$에는 가장 큰 가중치(0.5)를 부여하여 여유가 큰 경로 선택을 유도함으로써 핫스팟 분산을 주된 목표로 삼았다. 반면 지연 $d$, 홉 수 $h$, 경로 중복 $o$는 전송 비용과 자원 소모를 나타내는 비용 항이므로 음의 가중치(0.15, 0.1, 0.15)를 부여하되, 부하 분산과 충돌할 경우 양보하도록 상대적으로 작게 설정하였다.

$$\hat{\rho}=\frac{\rho}{\rho_{\max}},\quad \hat{d}=\frac{d}{d_{\max}},\quad \hat{h}=\frac{h}{h_{\max}},\quad \hat{o}=\frac{o}{h-1} \tag{4}$$

표 1. 보상 변수 정규화
Table 1. Reward-variable normalization

| Symbol | Variable | Normalizer | Range |
|---|---|---|---|
| ρ | Residual capacity | initial capacity (500) | [0, 1] |
| d | Propagation delay | max delay (280 ms) | [0, 1] |
| h | Hop count | max hops (22) | [0, 1] |
| o | Path overlap | path length (h-1) | [0, 1] |
| p | Bottleneck position | src = 0, dst = 1 | [0, 1] |

실패한 요청에는 성공 보상을 대체하는 강한 음의 페널티를 부여한다. 특히 용량이 소진된 경로에 대한 페널티는 병목 링크의 경로 내 위치 $p$에 따라 가중된다. $p$는 병목 링크가 경로상 어디에 있는지를 출발지를 0, 목적지를 1로 하여 나타낸 값으로, 병목이 도착에 가까울수록 그 지점에 도달하기까지 점유한 중간 링크 자원이 많아 낭비가 크다. 따라서 $p$가 클수록 페널티를 키워 막힐 경로라면 출발 측에서 일찍 차단되는 편이 유리하도록 유도하였다. 링크 장애와 그 외 실패에는 1.5의 페널티를 부여한다.

성공 보상의 최악값은 약 0.1로, 실패 페널티가 이보다 훨씬 크게 음수가 되어, 장애와 병목을 회피하는 것이 보상 구조를 지배하게 된다. 한편 식 (1)과 식 (2)가 요청 단위로 즉시 부여되는 보상인 것과 달리, 식 (3)의 종료 보상은 에피소드가 끝날 때 한 번만 가산되는 항이다.

N은 한 에피소드 동안 전달에 성공한 요청의 총수로, 이에 비례하는 보상을 더함으로써 개별 요청의 즉시 보상만으로는 포착하기 어려운 에피소드 전체의 누적 처리량을 극대화하도록 유도한다. 끝으로 보상 신호는 입력 상태 채널과 정합되도록 설계하였다.

강화학습에서 에이전트는 관측 가능한 정보에 대해서만 회피 정책을 학습할 수 있으므로, 보상에서 처벌하는 요인이 상태에 표현되어 있어야 한다. 이에 따라 병목 용량과 관련된 보상 항(ρ, p)은 경로별 병목을 나타내는 병목 채널인 채널 15 - 18로, 링크 장애 페널티는 장애 링크 여부를 나타내는 장애 채널인 채널 10으로 입력에 명시적으로 제공하였다. 반대로 이러한 채널이 없는 모델은 동일한 보상을 받더라도 병목, 장애를 회피하기 어렵다. 이처럼 입력 채널과 보상 설계는 서로 맞물려 동작하도록 구성하였다.

## 3.2 CNN 표현

CNN 표현은 144개의 위성을 12×12 격자에 배치한 텐서를 입력으로 받는다. 입력 채널 19개는 표 2와 같이 목적지, 출발지 표시, 정규화한 위도·경도, 북·남·동·서 방향별 링크 잔여 용량, 배터리, 요청 진행도, 링크 장애 indicator로 구성된 공유 채널 11개와 경로 노드 indicator 및 경로별 병목 용량으로 구성된 경로별 채널 2개×K=8개로 이루어진다.

모델 구조는 3×3 합성곱 스템(19→64), 잔차 블록 3개, AdaptiveAvgPool, Dueling Head로 구성되며 총 파라미터 수는 약 496K이다. 출력은 K개의 경로별 Q값이다.

표 2. CNN 입력 채널
Table 2. CNN input channels

| Idx | Channel | Description |
|---|---|---|
| 0 | dst_marker | Destination one-hot |
| 1 | src_marker | Source one-hot |
| 2-3 | lat/lon | Normalized lat/lon |
| 4 - 7 | cap_NSEW | N/S/E/W link residual capacity |
| 8-9 | battery/req | Battery / Request progress |
| 10 | fail | Failed link indicator |
| 11 - 14 | path_indicator | Node indicator (per-path) |
| 15-18 | bottleneck | Per-path bottleneck |

## 3.3 GNN 표현

GNN 표현은 144개의 위성을 노드로 하는 그래프를 입력으로 받는다. 노드 특징 16개는 표 3과 같이 목적지, 출발 표시, 정규화한 위도와 경도, 방향별 링크 잔여 용량, 배터리, 요청 진행도, 경로별 노드 표시, 실패 링크 표시의 기본 12개에 경로별 병목 용량, 타임스텝 내 부하율과 부하 추세, 사용량 EMA의 보조 특징 4개를 더해 구성되며, 정규화 인접 행렬을 메시지 패싱의 가중치로 사용한다.

표 3. GNN 노드 피처
Table 3. GNN node features

| Idx | Feature | Description |
|---|---|---|
| 0 | dst_marker | Destination one-hot |
| 1 | src_marker | Source one-hot |
| 2 | lat | Normalized latitude |
| 3 | lon | Normalized longitude |
| 4 – 7 | cap_NSEW | N/S/E/W link residual capacity |
| 8 – 9 | battery/req | Battery / Request progress |
| 10 | path_indicator | Node indicator (per-path) |
| 11 | fail | Failed link indicator |
| 12 | bottleneck | Per-path bottleneck capacity |
| 13 | load | Within-timestep load ratio |
| 14 | trend | Inter-timestep load trend |
| 15 | usage_ema | Usage EMA (history) |

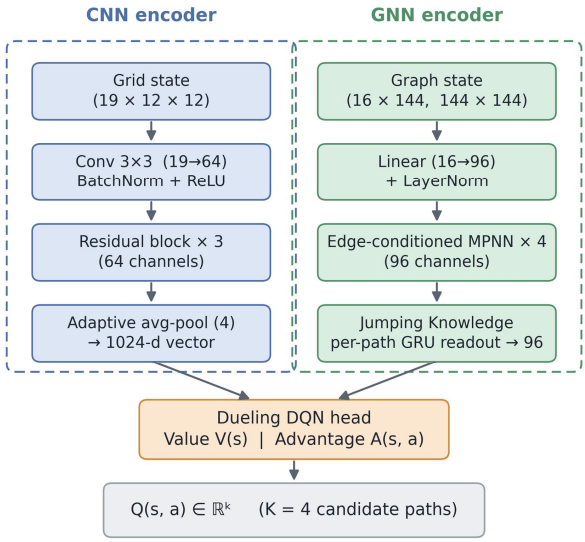



그림 3. CNN과 GNN 아키텍처
Fig. 3. CNN and GNN architectures

모델 구조는 Linear(16→96), LayerNorm, Edge MPNN 블록 4개, Jumping Knowledge 결합(층별 임베딩 5개를 480→96으로 사영), 전역 평균 풀링과 경로별 GRU readout, Dueling Head로 구성되며 총 파라미터 수는 약 430K이다. 출력은 CNN과 마찬가지로 K개의 경로별 Q 값이다. CNN이 경로별 병목 용량을 별도의 경로 채널로 입력하는 것과 달리, GNN은 동일 정보를 각 노드의 병목 특징으로 부여하여 메시지 패싱을 통해 이웃 노드로 전파한다. 구체적으로 CNN은 K개 경로에 각각 대응하는 전용 병목 채널인 채널 15 – 18에서 병목 값을 직접 읽지만, GNN은 병목을 노드 단위 특징으로만 부여하고 경로별 표현은 메시지 패싱과 경로별 GRU readout이 해당 경로가 지나는 노드들의 병목 정보를 집계하여 형성한다. 그림 3은 CNN과 GNN의 전체 아키텍처를 보여준다.

## Ⅳ. 성능 평가

### 4.1 시뮬레이션 환경

전체 1,081 타임스텝은 학습 분포 75%(810 타임스텝)와 홀드아웃 분포 25%(271 타임스텝)로 분할하였으며, 학습 분포로 모델을 학습한 뒤 두 분포에서의 성능을 각각 측정함으로써 미관측 토폴로지에 대한 일반화 성능을 분리하여 평가하였다. 트래픽 요청은 영역에 집중되도록 출발지 위성과 목적지 위성을 가중 샘플링하여, 균일 트래픽 가정을 넘어 실제 망에서 발생하는 부하 편중 상황을 반영하도록 구성하였다. 또한 CNN과 GNN은 동일한 사전 계산 경로 집합, 보상 함수, 학습 하이퍼파라미터(표 4)로 학습·평가하여, 표현 방식 이외의 요인을 통제하고 비교의 공정성을 확보하였다.

학습 안정성을 확보하기 위해 Double DQN[12], Dueling Network[13], 우선순위 경험 재생(PER)[14]을 결합한 표준 구조를 채택하였다. Double DQN은 가치 함수의 과대추정을 완화하고, Dueling 구조는 상태 가치와 행동 이점을 분리하여 학습 효율을 높이며, PER은 학습 가치가 높은 경험을 우선적으로 샘플링한다. 시뮬레이터는 PyTorch 기반으로 자체 구현하였으며, 모든 실험은 NVIDIA RTX GPU 1대에서 수행하였다. 또한 타깃 네트워크는 soft target update 방식으로 갱신하며, 핵심 학습 하이퍼파라미터는 표 4와 같다.

표 4. 학습 하이퍼파라미터
Table 4. Training hyperparameters

| Item | Value |
|---|---|
| Total iterations | 500 |
| Train ep / Eval ep | 5 / 3 |
| Batch / Replay buffer | 64 / 30,000 |
| Discount $\gamma$ / soft $\tau$ | 0.97 / 0.005 |
| Learning rate (cosine) | 2e-4 → 1e-5 |
| $\varepsilon$ decay / min | 0.9965 / 0.03 |
| PER $\alpha$ / $\beta$ | 0.6 / 0.4→1.0 |

학습 모델인 CNN과 GNN은 세 가지 휴리스틱 기준선과 비교한다. Shortest는 K개의 후보 경로 중 가장 짧은 경로를, Random은 매 요청마다 K개의 후보 경로를 균일 확률로 선택한다. MaxMin은 K개의 후보 경로 중 병목 링크의 잔여 용량이 가장 큰 경로를 선택한다. 세 방식은 학습이 필요 없으며 동일한 사전 계산 경로 집합을 사용하므로, 학습 기반 모델의 성능 향상이 표현 학습에서 기인하는지를 분리하여 확인할 수 있다.

## 4.2 성능 평가

본 장에서는 4.1에서 정의한 시뮬레이션 환경을 기반으로 CNN과 GNN 표현의 성능을 세 가지 측면에서 분석한다. 먼저 처리량을 통해 두 표현이 라우팅 정책으로서 어느 정도의 성능을 달성하는지 비교하고, 학습 이전 단계의 행동 선택 분포를 통해 두 표현이 갖는 귀납 편향(Inductive bias)의 차이를 직접 관찰한다. 마지막으로 추론 지연과 모델 크기를 측정하여 실제 운용 환경에서의 효율 측면을 평가한다.

### 4.2.1 학습 수렴

성능을 비교하기에 앞서, 두 모델이 학습을 통해 안정적으로 수렴하였는지 확인하였다. 그림 4는 학습 진행에 따른 보상의 수렴 여부를 나타낸다. 두 모델 모두 학습 초기에는 낮은 보상에서 출발하였으나, 학습이 진행됨에 따라 보상이 꾸준히 상승하여 약 500회 반복 이후 평탄한 구간에 도달하였다. 이는 두 모델 모두 라우팅 정책을 안정적으로 학습 가능함을 의미한다. 한편 에피소드 누적 보상은 실패 페널티의 영향으로 음의 값을 가지며, 0에 가까울수록 우수함을 의미한다.

수렴 이후 GNN은 CNN보다 약 25% 높은 보상 수준에 도달하였다. 이 보상 격차는 뒤에서 제시하는 처리량 격차보다 다소 큰데, 이는 보상이 전달 성공량뿐 아니라 링크 장애·병목 회피와 경로 효율에 대한 페널티까지 함께 반영하기 때문이다. 즉 GNN은 더 많은 요청을 전달할 뿐 아니라 장애와 혼잡을 더 효과적으로 회피하여, 처리량 단독 지표보다 종합 보상에서 더 큰 우위를 보인다.

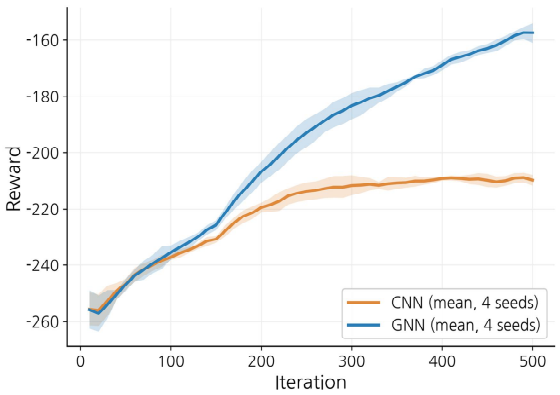



그림 4. 학습 진행에 따른 보상 수렴 곡선
Fig. 4. Reward convergence over training iterations

### 4.2.2 처리량

표 5는 출발지 위성 주위에 트래픽이 집중되는 상황에서 서로 다른 4개의 학습 시드로 재학습한 모델을 각각 100개 에피소드에서 측정한 에피소드당 평균 처리량을 의미한다. CNN은 174.8, GNN은 201.3으로 측정되어 GNN이 26.5 더 높은 처리량을 보였다. 두 모델 모두 휴리스틱 기준선 대비 일관되게 높은 처리량을 달성하였다. GNN은 4개의 모든 시드에서 일관된 처리량 우위를 보였으며, 동일 에피소드 기준 paired t-test에서도 통계적으로 유의한 차이를 나타냈다($p < 0.001$).

표 5. 100 에피소드 평균 처리량
Table 5. Mean throughput (Korea, 100 ep)

| Algorithm | Throughput (mean±std) | vs MaxMin |
|---|---|---|
| Shortest | 123.4 ± 32.6 | 25.3% |
| Random | 156.1 ± 40.6 | 5.6% |
| MaxMin | 165.3 ± 43.7 | – |
| CNN | 174.8 ± 46.4 | +5.8% |
| GNN | 201.3 ± 53.3 | +21.8% |

그림 5는 학습 진행에 따른 처리량 변화를 보여준다. CNN은 약 193, GNN은 약 221까지 안정적으로 수렴하였다. 또한, 학습이 실제로 성능 향상에 기여했는지 검증하기 위해, 그림 6에서 무작위 초기

화 모델과 학습된 모델을 동일한 30개 평가 에피소드에서 비교하였다. 그 결과 CNN은 115.1에서 175.3으로(+60.2), GNN은 120.0에서 202.4로(+82.4) 처리량이 향상되었다. 무작위 초기화 시점에서는 두 모델의 처리량이 유사했으나 학습 후 GNN의 향상 폭이 더 컸으며, 이는 GNN이 더 나은 표현력을 가짐을 시사한다.

그림 7은 학습 분포와 홀드아웃 분포에서의 처리량을 비교한 결과이다. CNN은 학습 192에서 평가 175로 약 9% 감소하였고, GNN은 학습 220에서 평가 201로 약 8% 감소하였다. 두 모델 모두 학습 대비 평가에서 일정 폭의 성능 하락이 관찰되었으나 그 비율이 유사하므로, GNN의 처리량 우위가 학습 분포에 과적합 된 결과가 아니라 미관측 토폴로지에서도 일관되게 유지됨을 확인할 수 있다.

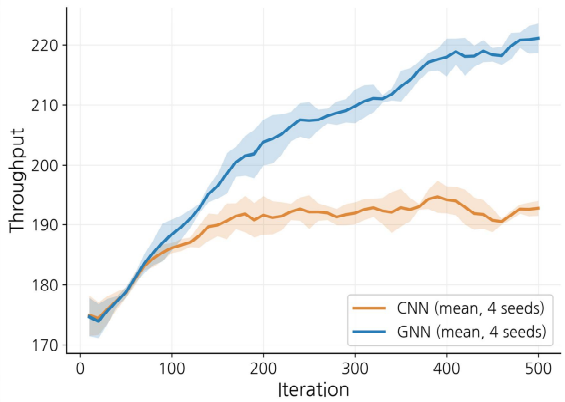



그림 5. 학습 진행에 따른 처리량 변화
Fig. 5. Throughput evolution over training iterations

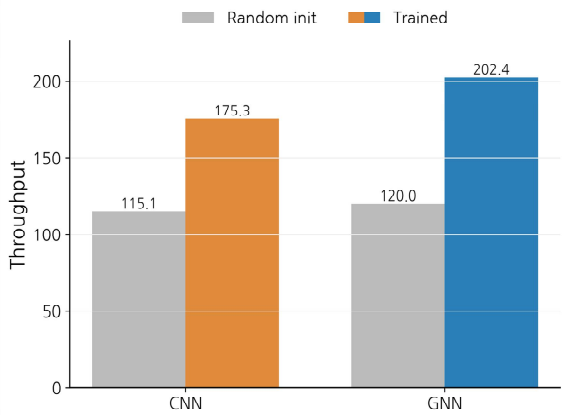



그림 6. 학습 전후 처리량 비교 (30 에피소드 평균)
Fig. 6. Throughput comparison before and after training (30-episode average)

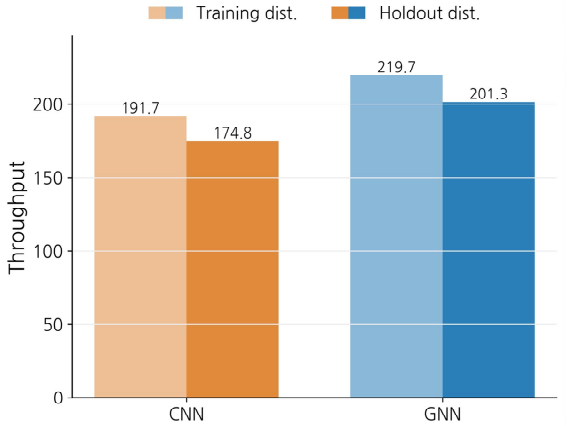



그림 7. 학습 분포와 홀드아웃 분포의 처리량 비교
Fig. 7. Throughput comparison between training and holdout distributions

#### 4.2.3 초기 행동 분포 분석

CNN과 GNN은 서로 다른 표현 구조를 가지므로, 학습 이전 단계에서도 서로 다른 초기 행동 선택 구조를 나타낼 수 있다. 이러한 구조적 차이가 행동 선택에 미치는 영향을 분석하기 위해 무작위로 초기화한 두 모델의 행동 선택 분포 엔트로피를 측정하였다.

표 6. 행동 선택 분포와 엔트로피
Table 6. Action-selection distribution and entropy

| Model | Path 0 | Path 1 | Path 2 | Path 3 | Entropy (bit) |
|---|---|---|---|---|---|
| CNN | 0.0000 | 0.0004 | 0.0000 | 0.9996 | 0.008 ± 0.012 |
| GNN | 0.4094 | 0.3727 | 0.0053 | 0.2126 | 1.579 ± 0.239 |

표 6은 무작위 초기화 상태에서 두 모델이 선택한 행동의 분포와 엔트로피를 비교한 결과이다. CNN의 엔트로피는 H ≈ 0.008 bit(4시드 평균)로 균등 분포(K=4, 최대 2 bit)의 약 0.4% 수준에 머물러 사실상 한 경로에 행동이 집중되었다. 반면 GNN은 H = 1.58 bit로 균등 분포의 약 79% 수준을 유지하여 학습 이전부터 여러 경로에 고르게 분산된 행동 선택 경향을 보였다. 다만 이 분포는 초기화 시점의 구조적 특성으로, 학습 중 탐색은 ε-greedy 정책(초기 ε=1.0의 균등 무작위)으로 별도

제어된다. 따라서 본 지표는 학습 데이터 수집 분포가 아니라 아키텍처가 K개 경로를 분리하는 능력을 나타내는 것으로 해석한다.

이 차이는 두 표현이 K개 경로를 분리하는 방식에서 비롯된다. CNN은 모든 경로에 대해 동일한 입력 텐서(144개 위성의 격자 표현)를 공유하며, 후보 경로 간의 구분은 출력 단계의 마지막 선형 계층 가중치만으로 이루어진다. 즉, 무작위 초기화 상태에서는 K개 경로에 대응하는 출력 가중치 중 어느 하나가 우연히 더 큰 값으로 초기화되면, 입력 상태와 무관하게 항상 동일한 경로의 Q-값이 가장 크게 산출된다. 그 결과 모든 상태에서 같은 경로가 선택되어 행동 분포가 한쪽으로 치우치게 된다. 반면 GNN은 경로 마스크를 통해 각 경로가 지나는 노드 집합을 별도로 입력받으므로, 경로마다 서로 다른 부분 그래프 위에서 함수를 평가한다. 이로 인해 출력 가중치가 무작위로 초기화되더라도 경로별 입력 자체가 다르므로 Q-값이 자연스럽게 분산된다.

즉 경로를 구분하는 능력이 GNN에서는 아키텍처 자체에 내재하는 반면, CNN에서는 학습을 통해서만 획득할 수 있다. 무작위 초기화 단계에서 나타난 이 차이는 GNN이 갖는 귀납 편향의 효과를 뒷받침하는 결과로 해석할 수 있다.

#### 4.2.4 추론 지연 및 모델 크기

라우팅 정책이 실시간 의사결정에 적용되기 위해서는 처리량뿐 아니라 추론 단계의 연산 효율도 함께 고려되어야 한다. 이를 정량적으로 비교하기 위해 CNN과 GNN의 평균 추론 지연과 모델 파라미터 수를 측정하였다.

추론 지연은 에피소드 전체 시간이 아니라 단일 요청에 대한 라우팅 결정 1회의 응답 시간을 측정하였다. 측정 과정에서 초기 컴파일 오버헤드의 영향을 배제하기 위해 워밍업 단계를 거친 후 1,000회 반복 측정하였으며, 결과의 안정성을 위해 평균과 분위수를 함께 산출하였다. 시간 측정은 GPU 환경에서 CUDA 이벤트 동기화를 사용하여 ms 단위로 기록하였다.

표 7은 동일한 입력에 대한 평균 추론 지연과 모델 크기를 비교한 결과이다. CNN의 평균 추론 지연은 2.84 ms, GNN은 6.46 ms로 CNN이 약 2.3배 빠르게 측정되었다. 모델 파라미터 수는 CNN 496K, GNN 430K로 큰 차이가 없으나, GNN은 매 추론 단계마다 144×144 인접 행렬에 대한 메시지 패싱 연산을 반복 수행하므로 실제 연산량과 지연이 CNN보다 크게 나타난다. 즉 모델 크기가 비슷하더라도 표현 구조에 따라 실측 추론 비용이 달라질 수 있음을 의미한다.

표 7. 추론 지연 및 파라미터
Table 7. Inference latency and parameters

| Model | Latency (ms) | Params |
|---|---|---|
| CNN | 2.84 | 496K |
| GNN | 6.46 | 430K |

이러한 결과는 표현 구조가 처리량뿐 아니라 추론 효율에도 영향을 미침을 보여준다. 처리량 측면에서는 GNN이 우위를 보였으나, 추론 효율 측면에서는 CNN이 약 2.3배 빠른 응답성을 보여 두 표현 간에 trade-off가 존재함을 확인할 수 있다.

전체적인 실험 결과는 CNN과 GNN의 처리량-속도 트레이드오프 관계를 보여준다. GNN은 15.2%의 처리량 우위를 가지나, CNN은 약 2.3배 빠른 추론 속도를 보여 정적 입력 형태로 FPGA 가속에 구조적으로 유리할 수 있다. 휴리스틱(MaxMin) 대비 CNN +5.8%, GNN +21.8%로 두 학습 모델 모두 의미 있는 개선을 보인다. 이러한 결과는 위성 라우팅을 위한 심층 강화학습 모델의 설계가 단일 기준이 아니라 운용 환경의 자원 제약과 응답 시간 요구를 함께 고려해야 함을 시사한다.

지상국 컨트롤러와 같이 연산 자원이 비교적 풍부한 환경에서는 처리량 우위를 갖는 GNN 표현이 적합한 반면, 위성 탑재 컴퓨터와 같이 전력, 메모리, 지연이 제약되는 환경에서는 정적 입력 구조와 짧은 추론 지연을 갖는 CNN 표현이 보다 실용적인 대안이 될 수 있다.

## Ⅴ. 결론 및 향후 과제

본 연구에서는 동일한 DQN 환경에서 격자형 CNN 표현과 그래프형 GNN 표현을 동일한 학습

및 평가 절차로 비교 분석하였다. 부하 집중 시나리오 100 에피소드의 평균 처리량 측정 결과, GNN은 CNN보다 약 15.2% 높았으나, 추론 지연은 CNN이 2.84 ms로 GNN보다 약 2.3배 짧게 측정되었다. 또한 무작위 초기화 상태에서의 행동 선택 분포 분석에서 GNN이 CNN보다 높은 엔트로피를 보였으며, 이는 GNN의 처리량 우위가 단지 더 큰 학습 용량 때문이 아니라, 경로를 구분하는 구조적 귀납 편향이 학습과 결합되어 나타난 결과임을 시사한다.

결론적으로 GNN은 그래프 구조를 직접 반영하는 표현 특성으로 인해 더 높은 처리량을 달성하였으며, 무작위 초기화 단계에서도 여러 경로에 분산된 행동 선택 경향을 보였다. 반면 CNN은 상대적으로 낮은 처리량을 보였으나 추론 지연이 짧고 구현이 단순하여 실시간 의사결정이 중요한 환경에 적합하였다. 이러한 결과는 저궤도 위성 네트워크에서 상태 표현 방식이 심층 강화학습 기반 라우팅 성능에 중요한 영향을 미치며, 네트워크 요구사항에 따라 적절한 표현 모델을 선택해야 함을 시사한다.

향후 연구에서는 CNN의 표현력을 향상시키기 위해 네트워크 상태 정보를 효과적으로 반영하는 다양한 채널 엔지니어링 기법을 적용하고, 이를 통해 처리량과 추론 효율 간의 절충 관계를 추가적으로 분석할 예정이다. 또한 CNN과 GNN의 특성을 결합한 GCN 기반 표현 모델을 대상으로 성능 및 추론 지연을 비교함으로써, 저궤도 위성 네트워크에 보다 적합한 상태 표현 방식을 탐색하고자 한다. 아울러 본 연구의 결과는 144개 위성, 12×12 격자, 부하 집중 시나리오라는 특정 실험 조건에서 도출된 것이므로, 향후 다른 규모의 위성 군집과 토폴로지, 트래픽 분포로 실험을 확장하여 일반화 가능성을 추가로 검증하고자 한다.

## References


[1] C. Westphal, L. Han, and R. Li, "LEO satellite networking relaunched: Survey and current research challenges", ITU J. Future Evolving Technol., Vol. 4, No. 4, pp. 711-744, Dec. 2023. https://doi.org/10.52953/lwxc19]28.

[2] X. Cao, Y. Li, X. Xiong, and J. Wang, "Dynamic routings in satellite networks: An overview", Sensors, Vol. 22, No. 12, Art. no. 4552, Jun. 2022. https://doi.org/10.3390/s22124552.

[3] C. Zhang, Q. Chen, Z. Tang, J. Wei, and G. Liu, "Pre-coded inter-satellite routing algorithm with load balancing for mega-constellation networks", Space: Sci. Technol., Vol. 4, Art. no. 0103, May 2024. https://doi.org/10.34133/space.0103.

[4] H. Wang, Y. Ran, L. Zhao, J. Wang, J. Luo, and T. Zhang, "GRouting: Dynamic routing for LEO satellite networks with graph-based deep reinforcement learning", Proc. 4th Int. Conf. Hot Information-Centric Networking (HotICN), Nanjing, China, pp. 123-128, Nov. 2021. https://doi.org/10.1109/HotICN53262.2021.9680855.

[5] Y. Shi, W. Wang, X. Zhu, and H. Zhu, "Low Earth orbit satellite network routing algorithm based on graph neural networks and deep Q-network", Applied Sciences, Vol. 14, No. 9, Art. no. 3840, May 2024. https://doi.org/10.3390/app14093840.

[6] F. Lozano-Cuadra, B. Soret, I. Leyva-Mayorga, and P. Popovski, "Continual deep reinforcement learning for decentralized satellite routing", IEEE Trans. Commun., Vol. 73, No. 10, pp. 8996-9012, Oct. 2025. https://doi.org/10.1109/TCOMM.2025.3562522.

[7] X. Liu, T. Ma, X. Qin, H. Zhou, and L. Zhao, "A DRL empowered multipath cooperative routing for ultra-dense LEO satellite networks", Proc. IEEE GLOBECOM, Kuala Lumpur, Malaysia, pp. 5961-5966, Dec. 2023. https://doi.org/10.1109/GLOBECOM54140.2023.10436959.

[8] E. Rapuano, et al., "An FPGA-based hardware accelerator for CNNs inference on board satellites: Benchmarking with Myriad 2-based solution for the CloudScout case study", Remote Sensing, Vol. 13, No. 8, Art. no. 1518, Apr. 2021. https://doi.org/10.3390/rs13081518.

[9] P. Almasan, J. Suárez-Varela, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio, "Deep

reinforcement learning meets graph neural networks: Exploring a routing optimization use case", Comput. Commun., Vol. 196, pp. 184-194, Dec. 2022. https://doi.org/10.1016/j.comcom.2022.09.029.

[10] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach", Proc. NIPS (Adv. Neural Inf. Process. Syst.), Denver, Colorado, USA, Vol. 6, pp. 671-678, Nov. 1993.

[11] V. Mnih, et al., "Human-level control through deep reinforcement learning", Nature, Vol. 518, No. 7540, pp. 529-533, Feb. 2015. https://doi.org/10.1038/nature14236.

[12] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning", Proc. AAAI, Phoenix, Arizona, USA, Vol. 30, No. 1, pp. 2094-2100, Feb. 2016. https://doi.org/10.1609/aaai.v30i1.10295.

[13] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning", Proc. AAAI, New Orleans, Louisiana, USA, Vol. 32, No. 1, pp. 3215-3222, Feb. 2018. https://doi.org/10.1609/aaai.v32i1.11796.

[14] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey", IEEE Signal Process. Mag., Vol. 34, No. 6, pp. 26-38, Nov. 2017. https://doi.org/10.1109/MSP.2017.2743240.

[15] P. Zuo, C. Wang, Z. Wei, Z. Li, H. Zhao, and H. Jiang, "Deep reinforcement learning based load balancing routing for LEO satellite network", Proc. IEEE 95th Veh. Technol. Conf. (VTC2022-Spring), Helsinki, Finland, pp. 1-6, Jun. 2022. https://doi.org/10.1109/VTC2022-Spring54318.2022.9860582.

[16] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications", AI Open, Vol. 1, pp. 57-81, 2020. https://doi.org/10.1016/j.aiopen.2021.01.001.

[17] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks", IEEE Trans. Neural Netw. Learn. Syst., Vol. 32, No. 1, pp. 4-24, Jan. 2021. https://doi.org/10.1109/TNNLS.2020.2978386.

[18] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning", Proc. AAAI, New Orleans, Louisiana, USA, Vol. 32, No. 1, pp. 3538-3545, Feb. 2018. https://doi.org/10.1609/aaai.v32i1.11604.

[19] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data", IEEE Signal Process. Mag., Vol. 34, No. 4, pp. 18-42, Jul. 2017. https://doi.org/10.1109/MSP.2017.2693418.

## 저자소개

김 범 석 (Bumseok Kim)

2025년 2월 : 국립금오공과대학교 소프트웨어공학과(학사)
2025년 3월 ~ 현재 : 국립금오공과대학교 대학원 소프트웨어공학과 석사과정
관심분야 : 그래프 신경망, 강화학습

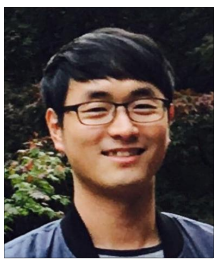
노 봉 수(BongSoo Roh)

2004년 2월 : 한양대학교 전자전기공학부(공학사)
2006년 2월 : 포항공과대학교 컴퓨터공학과(공학석사)
2021년 8월 : 충남대학교 컴퓨터공학과(공학박사)
2006년 4월 ~ 현재 : 국방과학연구소 책임연구원
관심분야 : 네트워크, 인공지능, 다층위성

한 명 훈 (Myoung-Hun Han)

2007년 2월 : 중앙대학교
컴퓨터공학(공학사)
2009년 8월 : 중앙대학교
컴퓨터공학(공학석사)
2021년 8월 : 중앙대학교
컴퓨터공학(공학박사)
2014년 10월 ~ 현재 :
국방과학연구소 선임연구원
관심분야 : 네트워크, M&S, LEO, 인공지능

이 헌 철 (Heoncheol Lee)



2006년 8월 : 경북대학교
전자전기컴퓨터학부(공학사)
2008년 8월 : 서울대학교
전기컴퓨터공학과(공학석사)
2013년 8월 : 서울대학교
전기컴퓨터공학과(공학박사)
2013년 9월 ~ 2019년 2월 :
국방과학연구소 선임연구원
2019년 3월 ~ 현재 : 금오공과대학교 전자공학부
IT융복합공학과 조교수
관심분야 : SLAM, 자율주행, 인공지능, 알고리즘 가속화

김 성 렬 (Sungryul Kim)

2010년 2월 : 부산대학교
컴퓨터공학과(학사)
2017년 8월 : 부산대학교 대학원
컴퓨터공학과(공학박사)
2019년 3월 ~ 현재 :
국립금오공과대학교
컴퓨터소프트웨어공학과 부교수
관심분야 : 빅데이터, 머신러닝