Doctoral Dissertation

AI-based NFV Management and Orchestration

Hee Gon Kim (김 희 곤)

Department of Computer Science and Engineering

Pohang University of Science and Technology

2024

AI 기반 NFV 관리 및 오케스트레이션

AI-based NFV Management and Orchestration

AI-based NFV Management and Orchestration

by

Hee Gon Kim
Department of Computer Science and Engineering
Pohang University of Science and Technology

A dissertation submitted to the faculty of the Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Computer Science and Engineering

Pohang, Korea
06. 14. 2024
Approved by
James Won-Ki Hong (Signature)
Academic advisor

AI-based NFV Management and Orchestration

Hee Gon Kim

The undersigned have examined this dissertation and hereby certify that it is worthy of acceptance for a doctoral degree from POSTECH

06. 14. 2024

Committee Chair	James Won-Ki Hong	(Seal)
Member	Young-Joo Suh	(Seal)
Member	Dongwoo Kim	(Seal)
Member	Inseok Hwang	(Seal)
Member	Seulbae Kim	(Seal)

DCSE 김 희 곤 Hee Gon Kim
20182237
AI-based NFV Management and Orchestration.
AI 기반 NFV 관리 및 오케스트레이션.
Department of Computer Science and Engineering ,
2024, 84p.
Advisor : James Won-Ki Hong.
Text in English.

ABSTRACT

The advent of Software-Defined Networking (SDN) and Network Functions Virtualization (NFV) has revolutionized the networking industry by transforming traditional network deployment, operations, and management. SDN enables centralized management by separating control and data forwarding functions, while NFV enhances network flexibility by virtualizing network services as software components called Virtual Network Functions (VNFs). The integration of these technologies facilitates rapid, dynamic resource allocation and service deployment, thereby maximizing network flexibility and management efficiency. However, despite these advancements, the increased complexity makes effective management more difficult. This thesis focuses on developing an AI-based NFV Management and Orchestration (NFV MANO) system designed to automate and optimize the management of VNFs. The proposed system incorporates various AI-driven NFV management functions, including VNF

deployment, auto-scaling, service function chaining, and consolidation. These functions are essential for effectively managing dynamically changing network environments. The primary research objective is to achieve a zero-touch network management system requiring no human intervention. To this end, the NFV-AI module within the system generates optimal policies for NFV management tasks through machine learning algorithms such as Graph Neural Networks (GNN) and Deep Reinforcement Learning (DRL). Additionally, the system includes a digital twin to replicate and simulate the NFV environment, providing a proactive learning environment for AI models. The evaluation of the proposed AI-based NFV MANO system demonstrates significant improvements in service quality, resource utilization, and operational efficiency. This research contributes to the field by presenting a comprehensive approach to integrating AI with NFV management, paving the way for fully autonomous network management systems.

Contents

List of Figures

I. Introduction

1.1 Motivation

In recent years, the networking industry has been revolutionized by paradigms such as Software-Defined Networking (SDN) and Network Functions Virtualization (NFV), transforming traditional network management. SDN separates the control and data forwarding functions of the network, allowing centralized management, as shown in Figure 1.1 [1]. This approach enables network administrators to efficiently manage the network from a single point, eliminating the need to configure multiple devices individually. On the other hand, NFV enhances network flexibility by virtualizing network services as software components called Virtual Network Functions (VNFs) that run on commercial servers, as shown in Figure 1.2 [2]. Combined with SDN, NFV facilitates rapid dynamic resource allocation and service deployment and maximizes network flexibility and management efficiency.

NFV Management and Orchestration (NFV MANO) refers to systems that manage and orchestrate network functions and services in an NFV environment. NFV MANO plays a crucial role in automating and optimizing VNF management by automatically deploying and configuring network services, thereby reducing installation

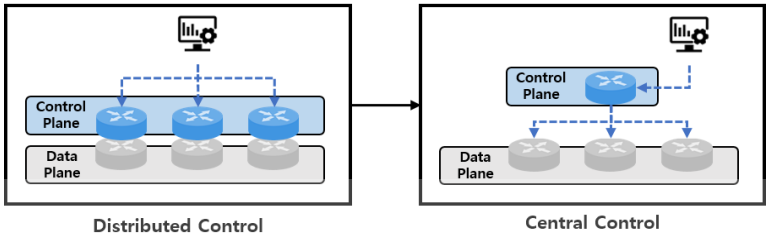

Figure 1.1: Software Defined Network

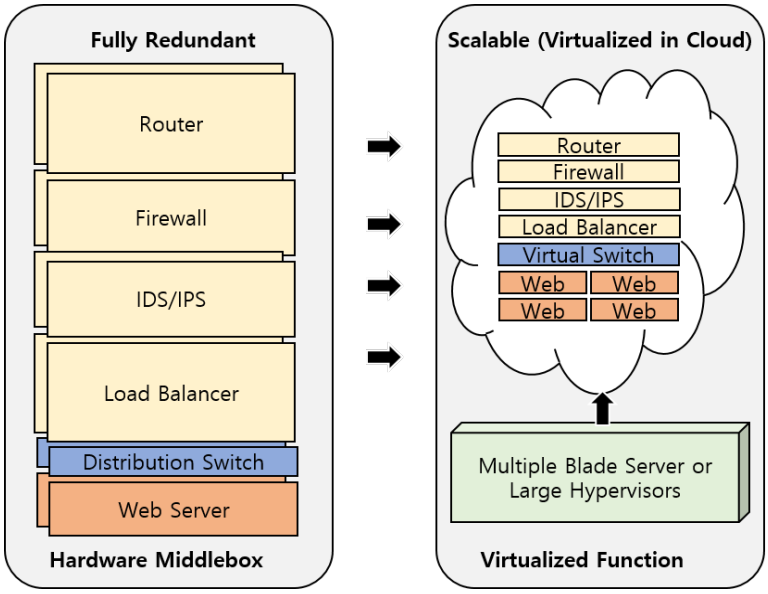

Figure 1.2: Virtual Network Function

times for new services and enhancing operational efficiency. Network services can be provided as Service Function Chains (SFCs), which sequentially deliver network traffic through a series of Network Services (NSs) [3]. In an NFV environment, SFCs are composed of a series of VNFs that provide specific network functions. The NFV MANO oversees network services at the SFC unit level, ensuring the efficient management and delivery of various user services. To manage SFCs effectively, the following NFV management functions are required:

- **VNF Deployment**: Optimizes the placement of VNFs on physical servers to maximize resource utilization and performance.
- **Auto-scaling**: Automatically adjusts the number of VNF instances in response to varying network demands to maintain performance and efficiency.
- **Service function chaining**: Optimizes routing paths and service quality by efficiently chaining multiple VNFs.

- **VNF Consolidation**: Reduces energy consumption by consolidating VNFs and optimizing resource usage across the network.

These NFV management functions are essential for optimally managing dynamically changing networks; however, creating optimal management policies for complex network environments is challenging and requires significant effort. As various services and applications demand high levels of automation and proactive decision-making, the complexity of NFV and the diverse virtualization stacks further complicate the situation. This complexity often renders traditional optimization techniques inadequate due to runtime constraints and hardware limitations. To address this issue, recent research has focused on using Artificial Intelligence (AI) technology to optimize each management function, with the ultimate goal of achieving Zero-touch network management, which requires no human intervention [4].

1.2 Problem Statement and Research Goal

To achieve a zero-touch network system, this thesis proposes an AI-based NFV-MANO system. The proposed system includes various AI-driven NFV management functions and an orchestration function that controls and coordinates the learning of each function. The main problem statement that this research addresses is summarized as follows:

- **NFV Management**: Generates optimal network management policies to enhance service quality while efficiently utilizing network resources based on the current network environment.
- **AI Management**: Oversees the learning of NFV management modules and ensures the continuous adaptation of optimal learning models to dynamically changing networks.
- **NFV Orchestration**: Automates NFV management functions, ensuring smooth

integration and coordination to prevent service quality degradation and operational inefficiencies due to conflicts among management functions.

- **NFV Digital Twin** [5]: Replicates and simulates the NFV environment, providing proactive learning environments for AI models.

Despite many attempts to apply artificial intelligence to network management, developing high-quality machine learning models for network management remains challenging due to the dynamic variability and complexity of network environments. Additionally, reinforcement learning-based models, primarily used for many VNF management tasks, are particularly vulnerable to environmental changes, requiring additional learning time and often performing poorly in changed network environments despite excellent performance during training. Furthermore, the various NFV management modules provided by NFV MANO can generate conflicting or competing management policies under specific circumstances. Failure to properly coordinate these management functions can degrade overall system performance. Therefore, this thesis addresses the essential elements required to resolve these issues. The major contributions of this research are as follows:

1. **AI-based NFV Management**: An NFV environment was established, and AI-based management policies were automated. The data collection, preprocessing, and performance evaluation of learning models were automated to proceed without human intervention. Different machine learning models were applied and trained for four VNF management policies (VNF deployment, service function chaining, auto-scaling, and VNF consolidation), demonstrating high performance.

2. **Compartmentalized Management Policies**: Various types of NFV management were compartmentalized to minimize conflicts among management functions. Network domain knowledge was utilized to reduce the complexity of management problems, lowering the difficulty for machine learning models.

3. **Proactive Learning System**: A system was developed to proactively conduct learning for various network environments. Appropriate pre-trained machine learning models were loaded for a real-time network environment to ensure optimal management.

4. **Orchestrated Management Policies**: The management policies of NFV management functions were orchestrated to avoid conflicts or competition among policies, thereby preventing system performance degradation.

5. **Digital Twin System**: A digital twin system was developed to replicate the NFV environment, creating a simulation environment for operating machine learning models. By learning through simulation models, it was possible to prevent actual testbed failures due to incorrect policies while enabling proactive learning for various network environments. Additionally, it provides various network environments to enable machine learning models to proactively learn about diverse conditions.

1.3 Organization

The remainder of this thesis is organized as follows. Chapter II presents background knowledge and related research to help readers understand this study. Chapter III defines the overall architecture of the proposed NFV MANO system, network environment information, and the problem definitions and solutions of the NFV Management module and Orchestration module. Chapter IV details the implementation elements of each component of the proposed NFV MANO and the neural network structures for machine learning. Chapter V presents evaluation results, showing the improved performance of the NFV Management modules through the proactive learning system and smooth coordination through the orchestration system. Chapter VI concludes with key observations and discussions on future research.

II. Background and Related Work

2.1 Network Function Virtualization

2.1.1 NFV MANO

NFV enables flexible and efficient network operations by implementing network services in software. To effectively manage NFV, the European Telecommunications Standards Institute (ETSI) has defined the NFV MANO framework as shown in Figure 2.1 [6]. The NFV MANO is a system responsible for the management and orchestration of NSs and VNFs in an NFV environment, consisting of several components as follows:

- **NFV Infrastructure (NFVI)**: NFVI provides the physical hardware and virtualized resources on which VNFs are executed. It uses hypervisors or container orchestration platforms to virtualize physical resources, create virtual networks, and provide network connections for communication between VNFs. By utilizing NFVI, the flexible deployment and resource utilization of VNFs are enhanced through a virtualized environment.
- **Virtualized Infrastructure Manager (VIM)**: The VIM manages the NFVI resources and continuously monitors the state of the NFVI. It allocates and releases resources assigned to the virtual network and configures and manages network connections between VNFs.
- **VNF Manager (VNFM)**: The VNFM manages the lifecycle of individual VNFs. It creates and initializes new VNF instances and manages the configuration of VNFs to ensure their correct operation.
- **NFV Orchestrator (NFVO)**: The NFVO manages the lifecycle of NSs. This

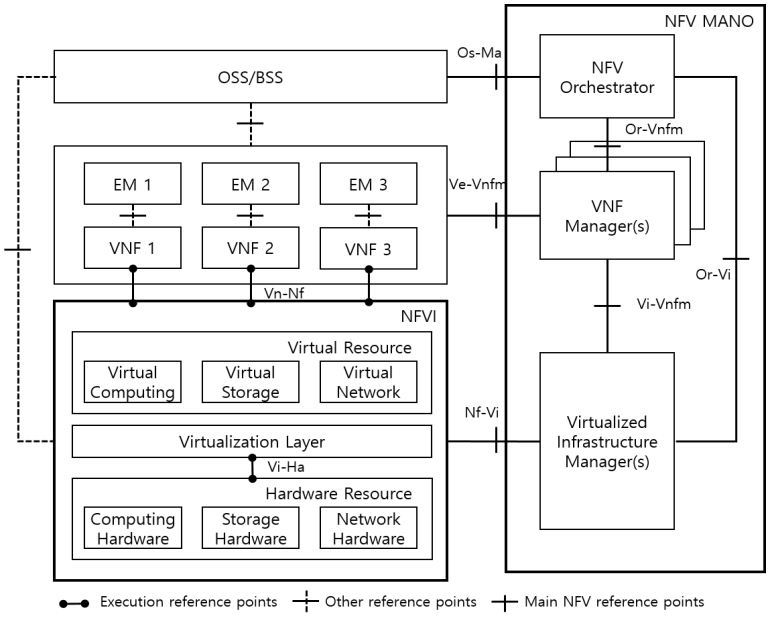

Figure 2.1: ETSI NFV MANO

includes onboarding instances, creation, removal, monitoring, scaling, and upgrading network services. It integrates new network services into the system and optimizes existing network services.

Network administrators monitor VNFs to enhance network service quality and ensure efficient network operation, implementing suitable management policies as required. The types of management policies vary depending on the network's purpose and type, but four key management policies are critical and challenging for integrated management. Their meanings and detailed functions are described in the following sections.

2.1.2 VNF Deployment

To provide network services, VNFs that deliver network functions must be prepared. Therefore, NFV MANO must (pre-)install VNFs according to the requirements

of specific network services. VNFs are provided in the form of virtual machines or containers, and are allocated resources such as CPU, memory, and disk using pre-configured templates. These templates define the necessary software for network function operation, including operating systems and file systems. The installed VNFs set up network and security groups to connect with internal and external networks. Afterward, SFCs utilize VNFs to deliver network services. By deploying VNFs on suitable physical servers in response to network service requirements, network administrators can effectively utilize resources while delivering high-quality services.

2.1.3 VNF Auto-scaling

VNF auto-scaling automatically adjusts the number of VNF instances to optimize performance and availability in an NFV environment. It dynamically allocates resources based on network traffic fluctuations to ensure network service stability and efficiency. When network traffic increases, auto-scaling expands resources to maintain network performance. Conversely, when traffic decreases, it reduces resources to save costs. This approach minimizes operational costs by using resources only when needed and enhances availability by providing stable network services even during sudden traffic changes.

Auto-scaling strategies include vertical scaling (scaling up/down) and horizontal scaling (scaling out/in). Vertical scaling adjusts the resources (CPU, memory) of existing VNF instances to control performance, whereas horizontal scaling increases or decreases the number of VNF instances to distribute the traffic load. Vertical scaling is simpler to manage and is beneficial for session-based applications, as it only adjusts the resources of existing instances without changing their state. However, it has limitations in scalability due to physical server resource limits and can experience slow recovery times if an instance fails. Conversely, horizontal scaling achieves high scalability and availability by adding or removing VNF instances. However, it involves the complexity of managing multiple instances, and the time required to deploy new

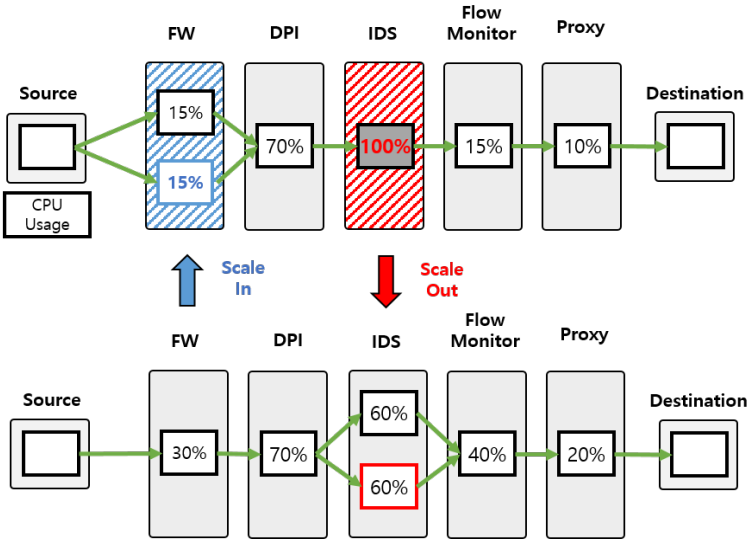

Figure 2.2: VNF Scaling (Scale in out)

instances. The choice or combination of scaling methods depends on the characteristics of the network service and NFVI. This thesis focuses on addressing scaling issues through horizontal scaling. While horizontal scaling presents complexities, we aim to mitigate them using AI. Figure 2.2 illustrates the process of horizontally scaling VNFs.

2.1.4 Service Function Chaining

Service function chaining determines the sequence of network services that data packets must pass through. Once the chain is established, the classifier routes data packets to specific service chains based on packet attributes such as IP addresses and port numbers. Figure 2.3 below illustrates an example of SFC operation, where appropriate VNFs are selected from already installed VNFs to form an optimal SFC. This approach prevents service quality degradation by selecting underutilized VNFs and forming chains that minimize service latency, ensuring the shortest path. For instance, in Figure 2.3, if the current service's destination is node 5, the chain is formed to node 5's proxy instead of node 2's proxy. This chaining process not only creates the initial

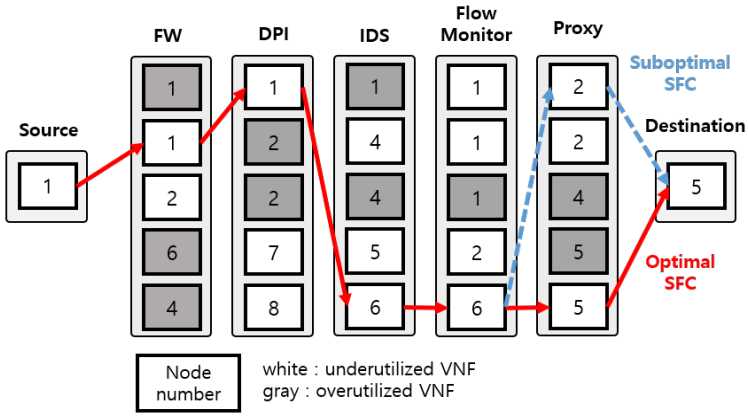

Figure 2.3: Service Function Chaining

SFC but also continuously reconfigures the chain routes by monitoring the quality of ongoing network services to prevent service quality degradation.

2.1.5 VNF Consolidation

VNF consolidation involves consolidating multiple VNF instances onto a reduced number of physical servers, aiming to optimize resource allocation and enhance operational efficiency. When resources are spread across multiple locations, managing them independently can hinder optimal resource usage and increase overall system operating costs. VNF Consolidation reduces unnecessary resource wastage and prevents resource fragmentation by integrating distributed resources. Also, by concentrating power consumption on fewer servers, this approach can achieve higher power efficiency.

2.2 Machine Learning

2.2.1 Graph Neural Network

Graph Neural Networks (GNNs) are artificial neural networks designed to learn patterns from graph data and understand complex relationships and interactions [7]. GNNs can be powerful tools for network management, effectively addressing issues such as traffic prediction, network optimization, and anomaly detection. They update each node's representation iteratively using node and edge information. The primary operation principles are as follows:

1. **Message Passing**: Each node receives messages from its neighboring nodes and edges to update its state.

2. **Aggregation**: Each node aggregates the received messages to calculate a new embedding, typically using operations such as sum, mean, or max.

3. **Updating**: Nodes update their states based on the aggregated messages, with parameters learned through neural networks.

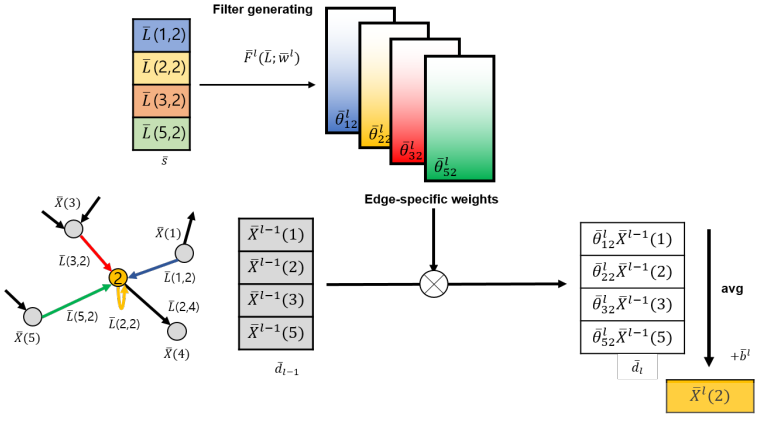

Figure 2.4: Edge-conditioned Convolution

GNNs are classified by the type of neural network used for learning, mainly into Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs), and Graph Recurrent Networks (GRNs). Figure 2.4 shows an example of a GCN model, edge-conditioned filtered graph convolutional neural network (ecGNN) [8]. In this model, each node receives information from adjacent nodes and averages it to create node and graph embeddings. During learning, edge information is converted into filters and multiplied with the passing messages.

2.2.2 Deep Reinforcement Learning

Reinforcement Learning (RL) is a type of machine learning where an agent learns to take optimal actions within an environment [9]. Unlike supervised learning, RL does not require labeled datasets or explicit answers. Instead, the agent interacts with the environment by observing the current state, taking actions, and receiving feedback in the form of rewards or penalties. Over time, the agent learns better policies and ultimately discovers an optimal policy or strategy that maximizes cumulative rewards.

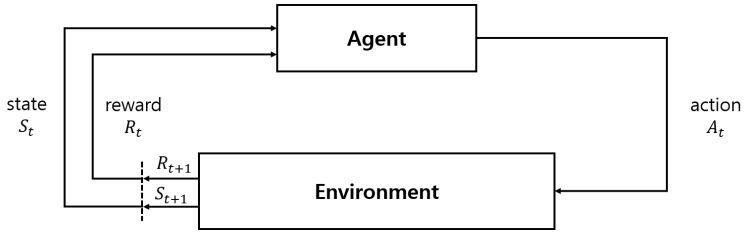

Figure 2.5: Generic Flow of Reinforcement Learning

An RL system primarily consists of two main entities: the agent and the environment. Figure 2.5 shows generic flow of RL. The agent observes the current state S_t in the environment and selects actions A_t based on this state and its policy. The action is executed in the environment, generating a new state S_{t+1} and a reward R_t for the agent's action. Through repeated interactions, the agent learns the optimal actions.

Deep Reinforcement Learning (DRL) combines RL algorithms with deep neu-

ral networks (DNNs) to represent and approximate policy or value functions[10]. DRL is particularly effective for complex and large-scale problems, handling high-dimensional state spaces in complex environments. In this study, we used the Deep-Q-Network (DQN) [11] algorithm and the Proximal Policy Optimization (PPO) [12] algorithm to train DRL models.

- **DQN**: DQN is a value-based RL algorithm that learns the state-action value function (Q-function). The Q-function represents the expected cumulative reward for a given state and action. Initially, Q-values are arbitrary or set to zero, but they are updated gradually by calculating the difference from the target value. The target value includes the expected reward obtained after taking an action in the current state and the maximum expected reward in the next state. By experiencing multiple state-action pairs, the Q-function improves. DQN uses deep neural networks to approximate the Q-function, effectively handling high-dimensional state spaces. It stores experienced data in memory and samples mini-batches randomly for learning, reducing data correlation and improving learning efficiency. Additionally, a target network is introduced to enhance learning stability. The target network is updated periodically with the policy network's weights, stabilizing the target values.

- **PPO**: PPO is a policy-based RL algorithm that directly learns policies representing the probability of taking actions given to states. Policy-based algorithms update policy parameters based on the gradient of the expected cumulative rewards. The Trust Region Policy Optimization (TRPO) algorithm limits the range of changes during policy updates to ensure stability and uses the Advantage function to calculate how much better specific actions are compared to the average, learning through an objective function [13]. Instead of using the Kullback-Leibler divergence [14] constraint to limit policy update changes as in TRPO, PPO uses a clipping technique, reducing implementation complexity and computational costs. However, sensitive hyperparameter tuning is required

for the clipping range.

DQN and PPO perform reinforcement learning using value-based and policy-based approaches, respectively. DQN is effective in discrete action spaces and is simple and intuitive to implement, but may perform poorly in continuous action spaces. Conversely, PPO performs well in continuous state and action spaces, ensuring stability through reliable policy updates. Both algorithms have their own advantages and are chosen based on the problem's characteristics and requirements.

2.3 Related Work

2.3.1 AI-based NFV Management

There have been several attempts to apply AI models to NFV (Network Function Virtualization) management. In AI-driven network management, Lange et al. [15] predicted VNF (Virtual Network Function) placement decisions in dynamically changing network states. The authors used Integer Linear Programming (ILP) to generate VNF placement solutions and then employed the H2O AutoML framework [16] to train various machine learning models such as XGBoost [17], GBM [18], DRF [19], XRT [20], and neural networks for VNF placement learning.

Lee et al. [21] proposed a Deep Q-Networks (DQN)-based method for auto-scaling VNFs. The authors defined a reward model considering node utilization and VNF instance density, developing a model that minimizes Service Level Agreement (SLA) violations while efficiently using resources. Heo et al. [22] introduced a GNN-based encoder-decoder learning model for service function chaining. This study used GNN to cope with changes in network topology and unforeseen conditions, employing an encoder-decoder structure to understand node connections and generate the shortest paths. Chen et al. [23] proposed the Optimal Placement and Chaining (OPC) algorithm, focusing on maximizing the placement of SFC instances at the network edge. The proposed algorithm uses a Deep Reinforcement Learning (DRL) model to evaluate

AI efficiency in SFC orchestration and addresses network scalability problems. They use a Parameterized Action Markov Decision Process (PAMDP) [24] to model dynamic network state transitions with various parameters and Quality of Service (QoS) requirements. Huang et al. [25] proposed the scalable Service Function Chain Orchestration (SSCO) solution incorporating Federated Reinforcement Learning (FRL). FRL integrates Federated Learning for global model training [26], and SSCO divides the network into regions, assigning local agents to train individual models, which are then aggregated globally. Jeong et al. [27] suggested a transformer-based DRL approach to enhance energy efficiency through VM consolidation. This research conducted the VM consolidation process in two stages, reducing energy consumption during low-traffic periods by relocating VNFs and maintaining network performance.

2.3.2 NFV Orchestration

Several open-source projects are dedicated to NFV orchestration, including OPNFV[1], OPEN-Baton[2], OPEN-MANO[3], Tacker[4] , Cloud4NFV [28], and vConductor [29]. OPNFV provides an open platform for NFV infrastructure integration and testing, incorporating various open-source projects (e.g., OpenStack [30], KVM [31], OpenDaylight [32]) to form the NFV infrastructure stack. ONAP[5] offers end-to-end network service automation for design and orchestration.

OPEN-Baton and OPEN-MANO comply with ETSI standards for NFV orchestration but face interoperability issues with specific VIMs. Tacker, integrated with OpenStack, provides VNF and network service management functions. Cloud4NFV and vConductor support automation in managing NFV infrastructure and services. These open-source projects primarily focus on basic NFV management aspects. For instance, they prevent VNF deletion while assigned to a specific SFC until the SFC is

[1] https://www.opnfv.org/
[2] https://open-baton.github.io/
[3] https://github.com/nfvlabs/openmano/wiki
[4] https://github.com/openstack/tacker
[5] https://www.onap.org/

removed.

The End-to-End SFC Orchestration (ETSO) [33] framework aims to orchestrate more complex network management, processing SFC requests using Topology and Orchestration Specification for cloud applications (TOSCA) dataset [34]. ETSO deploys VNFs optimally and configures SFCs with an intelligent placement module, supporting several plugins and algorithms like Greedy [35], Multi-Stage [36], Dynamic Programming (DP) [37], Monte Carlo Tree Search (MCTS) [38, 39], and Eigen [35].

ETSI's Experiential Network Intelligence (ENI) Industry Specification Group (ISG) applies AI to networks and proposes several methods for flexible network resource utilization. ENI defined an AI-based MANO architecture and metadata-based policies to adjust services based on network conditions and user demands [40]. AI-based approaches for network slicing and resource availability improvements were presented, emphasizing data analysis-based models. These models employ machine learning algorithms such as reinforcement learning (RL) for elastic schedulers and deep learning models for predicting signal-to-noise ratio (SNR) [41].

The Network Intelligence (NI) project [21], aligned with the ETSI standard, aims for zero-touch network management through autonomous AI learning. NI provides an AI module for VNF resource demand prediction, VNF deployment, and several SFC management modules within MANO. NI developed the NI-Mon function for data collection, preprocessing, and provision to facilitate MANO. This thesis improves on the NI research by enhancing NI-MON and NFV management module performance by developing an integrated orchestration platform.

III. System Design

In this chapter, we define the physical network and virtual network environments. Following this, we present the overall structure of the proposed NFV MANO system and introduce the detailed design of the components. We define the problems and list the attempted solutions for each AI-based NFV Management module.

3.1 Definitions

Let $P = (N_P, L_P)$ be the physical network, where N_P is the set of physical nodes and L_P is the set of physical links. Each physical node $n \in N_P$ and physical link $l \in L_P$has the following attributes:

$n\_cores(n)$: Total number of cores of node n

$n\_cores\_free(n)$: Available cores of node n

$ram\_mb(n)$: Total amount of RAM (mb) of node n

$ram\_free\_mb(n)$: Available RAM (mb) of node n

$disk(n)$: Available disk space (gb) of node n

$type(n)$: 1 If node n is compute node, else 0

$power(n)$: 1 If power of node n is power on, else 0

$id(n)$: ID of node n

$max\_bw\_mbps(l)$: Maximum bandwidth usage (mbps) of link l

$cur\_bw\_mbps(l)$: Current bandwidth usage (mbps) of link l

$delay\_us(l)$: Latency (us) of link l

Before defining the virtual network, we define flavor F. A flavor defines the resource profile required for a VNF to run. It specifies the characteristics of the resources the VNF needs, such as the number of CPU cores, amount of RAM, and disk space. For example, a flavor that meets these requirements will be assigned if a specific VNF needs 2 CPU cores, 2GB of RAM, and 10GB of disk space to operate smoothly. An image refers to the virtual machine image or container image that includes the software package and operating environment of the VNF. The image contains all the software, libraries, and configuration files needed to provide the specific VNF functionality. However, we assume the image to be dependent on the flavor, so only the flavor needs to be considered. Each flavor $f \in F$ has the following attributes:

$n\_cores(f)$: Number of CPU cores required by flavor f

$ram\_mb(f)$: Amount of RAM (mb) required by flavor f

$disk(f)$: Amount of disk space (gb) required by flavor f

$capacity\_mbps(f)$: Maximum bandwidth usage (mbps) of flavor f

$delay\_us(l)$: Average latency (us) of flavor f

$id(f)$: ID of flavor f

Let $V = (N_V, L_V)$ be the virtual network, where N_V is the set of VNFs and L_V is the set of virtual links between VNFs. Each VNF $v \in N_V$ and virtual link $e \in L_V$ has the following attributes:

$flavor(v)$: Flavor ID of VNF v

$node\_id(v)$: The physical node where VNF v is installed

$cur\_cpu(v)$: Current CPU usage of VNF v

$cur\_ram(v)$: Current RAM usage of VNF v

$cur\_disk\_free(v)$: Current available disk (gb) of VNF v

$id(v)$: ID of VNF v

$cur\_tx(e)$: Current transmitted packets of virtual link e

$cur\_rx(e)$: Current received packets of virtual link e

$cur\_dropped\_tx(e)$: Current dropped transmitted packets of virtual link e

$cur\_dropped\_rx(e)$: Current dropped received packets of virtual link e

We have defined network service requests as Service Function Chain Requests (SFCRs). The NFV MANO takes SFCRs as input and generates the appropriate SFCs. Let $B = (b_1, b_2, \ldots, b_m)$ and $C = (c_1, c_2, \ldots, c_k)$ are the set of SFCRs and SFCs, respectively. Each SFCR $b \in B$ and SFC $c \in C$ has the following attributes:

$source\_client(b)$: Source client of SFCR b

$destination\_client(b)$: Destination client of SFCR b

$nf\_chain(b)$: The chain of VNFs type for SFCR b

$sla(b)$: SLA threshold of SFCR b

$id(b)$: ID of SFCR b

$sfcr\_ids(c)$: Set of SFCR IDs requesting service for SFC c

$vii(c)$: Group set of VNF IDs providing function for SFC c

$id(c)$: ID of SFC c

Lastly, We have defined network flows. Let $T = \{t_1, t_2, \ldots, t_p\}$ be the set of traffic flows. Each traffic flow $t \in T$ has the following attributes:

snodeID(t) : Source node ID of traffic t

dnodeID(t) : Destination node ID of traffic t

bandwidth(t) : Bandwidth of traffic t

duration(t) : Duration of traffic t

nf_chain(t) : The chain of VNF types of traffic t

sfcrID(t) : SFCR ID associated with traffic t

sfcID(t) : SFC ID associated with traffic t

svnfID(t) : Source VNF ID of traffic t

dvnfID(t) : Destination VNF ID of traffic t

start_time(t) : Start time of traffic t

3.2 Overview of the AI-based NFV MANO

Figure 3.1 shows the architecture of the proposed AI-based NFV MANO. The system consists of an NFV-MON, NFV-AI, NFV-Twin, and various additional functionalities with ETSI's NFV MANO components. The monitoring module, NFV-MON, is used to access the status of physical and virtual resources, and network service information is received and stored in a time-series order. For efficient network management, a large amount of data is collected, processed, and delivered appropriate

data according to module requests.

Each NFV Management module in NFV MANO continuously interacts with the NFVI. Each module demands different operations and information from the NFVI. Due to the dynamic nature of network environments, new management modules may be added to fulfill the objectives of various network services. In the proposed NFVO MANO architecture, we designed VIM Function (VIMF) and VIM Interface (VIMI) modules to ensure compatibility between the NFVI and the management modules. The VIMF is connected with both the NFVI and the monitoring module, enabling it to support not only basic NFVI control functions but also advanced functionalities within NFV Management. The VIMI provides direct control over the operating system of VNFs within the NFVI and facilitates the management of functionalities at the software level.

NFV MANO requires a connection between NFVO, VNFM, and VIM, and each function must be able to deliver, delete, activate, and deactivate policies through various policy management interfaces to the NFVI. Utilizing OpenStack as the NFVI allows basic management actions of VNFs and SFCs to be delegated through the OpenStack Tacker system, which orchestrates and manages VNFs. However, because it does not ensure overall system optimization or maintenance of network service quality, the NFVO needs to generate policies for complex VNF management actions and deliver them to the VNFM for execution in the VIM.

The NFV-AI module generates optimal policies for various NFV management tasks required by the system through learning. It determines the lifecycle policies of VNFs and SFCs. The module refines network management policies and uses customized machine learning models to generate the best management policies. Additionally, the Learning Controller module oversees the NFV-AI module, generating learning models for diverse network environments and selecting the most suitable model for specific network conditions. Furthermore, the Policy Controller module prevents conflicts between NFV Management modules and ensures efficient orchestration overall.

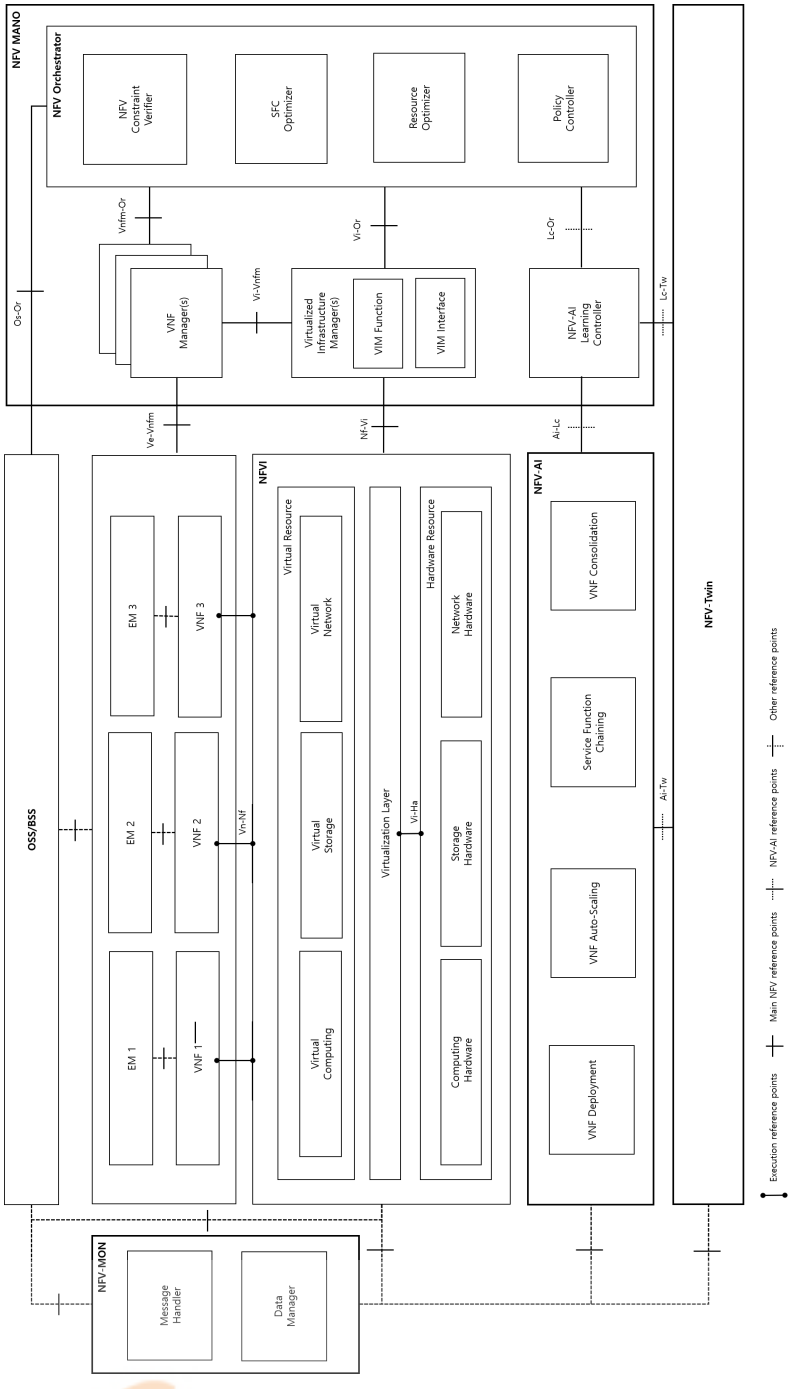

Figure 3.1: The architecture of the Proposed AI-based NFV MANO

3.3 NFV-Twin

The proposed system utilizes NFV-Twin, a replica of the NFV environment. NFV-Twin acts as a Network Digital Twin (NDT), replicating the attributes, behaviors, and operations of the actual network, allowing for a comprehensive analysis and simulation of the network environment. The NFV-Twin integrates various data sources, including real-time network statistics, and configuration information, to create a dynamic and interactive representation of the network.

Figure 3.2 illustrates the structure of NFV-Twin. NFV-Twin includes all objects provided by the NFVI (flavor, node, link, VNF, SFC, SFCR, traffic) and replicates all VIM functionalities. While the VIMF and VIMI, integrated with the existing VIM, operate directly connected to the actual testbed, the NFV-Twin also provides the same functionalities to the twin network. Since the twin network does not need to provide actual network functions, the twin-VIMI offers only a subset of functionalities.

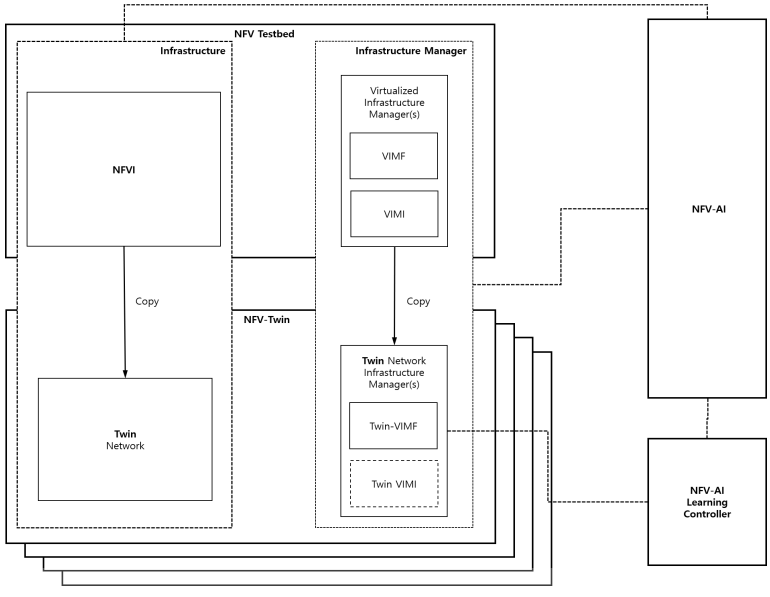

Figure 3.2: NFV-Twin

The NFV-Twin generates various twin networks under the control of the Learn-

ing Controller. It can create additional VNFs, SFCRs, SFCs, and traffics, and can intentionally induce load on VNFs. These generated twin networks are used for the NFV-AI to learn from diverse environments. This structure provides a stable learning environment by preventing testbed failures due to incorrect policies during the training of machine learning models such as reinforcement learning. Moreover, preemptive learning across various environments enables immediate deployment of the trained model in dynamic settings, eliminating the need for additional training.

3.4 NFV-AI

The proposed NFV MANO architecture optimizes and automates VNF deployment, auto-scaling, service function chaining, and consolidation. NFV-AI learns machine learning models for each function to generate optimal network management policies. Although these NFV management functions operate with different objectives, their goals and actions may overlap in certain areas. The proposed system orchestrates these overlapping parts to optimize the overall system. However, it would be much better to minimize policy conflicts by compartmentalizing the management functions from the beginning. Therefore, each NFV management function is defined with a precise scope to minimize potential conflicts between management tasks. Below are the broad and precise definitions for each NFV function.

- **Broad Definitions**:
  - Deployment: Optimization of VNF deployment, migration, and removal for **continuous** network service requests.
  - Auto-scaling: Optimization of chain configuration and scale-out/in VNF for **all deployed SFCs**.
  - Service function chaining: Optimization of chain configuration and **VNF placement** for **all deployed SFCs**.

  - Consolidation: Optimization of VNF migration and removal to reduce energy and fragmentation costs for the network.

- **Precise Definitions**:
  - Deployment: Optimization of VNF placement in response to a single network service request at **the current time**.
  - Auto-scaling: Optimization of chain configuration and VNF scale-out/in for **a selected SFC**.
  - Service function chaining: Optimization of chain configuration for **a selected SFC**.
  - Consolidation: Optimization of VNF migration to reduce energy and fragmentation costs.

By using precise definitions, policy conflicts between functions have been reduced. It reduces the overlapping areas between auto-scaling and service function chaining, and the overall network optimization problem has been simplified to focus on optimizing for a single problem rather than the entire network.

3.4.1 VNF Deployment

The VNF deployment function optimally places VNFs in response to network service requests. Depending on the type of service requested, various types of VNFs are deployed. The quality of network service is determined by the placement of these VNFs on physical servers. If VNFs are deployed without considering routing, delays can increase, and placing them on overly congested physical servers can result in constraints on bandwidth or network resource allocation. Therefore, to achieve optimal VNF placement, the appropriate types of VNFs must be arranged in the optimal order along the shortest path between the source node and the destination node of the requested service, and placed on an appropriate physical server considering network resources.

The optimal VNF deployment solution can be calculated using Integer Linear Programming (ILP). The following equation formulates the deployment problem. This equation defines the energy cost (Equation 3.1), traffic forwarding cost (Equation 3.2), SLO violation cost (Equation 3.3), and resource fragmentation cost (Equation 3.4), and generates the optimal VNF deployment method that minimizes these costs [42].

$$\mathbb{E} = \sum_{n \in N_P} \sum_{v \in N_V} (energy_{idle} + (energy_{peak} - energy_{idle}) cur\_cpu, ram(v)) \quad (3.1)$$

$$\mathbb{T} = \sum_{n \in N_P} \sum_{m \in N_P} (cur\_bw\_mbps(l_{n,m})), n \neq m \quad (3.2)$$

$$\mathbb{S} = \sum_{c \in C} \sum_{v \in \{\{id(v)\} \subseteq vii(c)\}} \max(delay\_us(f_v)(\sum_{i \in N_V} (cur\_tx(e_{vi})), 0) \quad (3.3)$$

$$\begin{aligned}
\mathbb{F}_{cpu} &= \sum_{n \in N_P} (n_cores_free(n) \text{ if } n_cores_free(n) \neq n_cores(n) \text{ else, } 0) \\
\mathbb{F}_{ram} &= \sum_{n \in N_P} (ram_mb_free(n) \text{ if } ram_mb_free(n) \neq ram_mb(n) \text{ else, } 0) \\
\mathbb{F}_{bw} &= \sum_{n \in N_P} \sum_{m \in N_P} (max_bw_mbps(l_{n,m})) - cur_bw_mbps(l_{n,m})) \\
&\quad \text{if } max_bw_mbps(l_{n,m})) \neq cur_bw_mbps(l_{n,m})) \text{ else, } 0)
\end{aligned} \quad (3.4)$$

The objective of ILP is minimizing $(\acute{a}\mathbb{E} + \acute{b}\mathbb{T} + \acute{c}\mathbb{S} + \acute{d}\mathbb{F})$ [43]. $\acute{a}$, $\acute{b}$, $\acute{c}$, $\acute{d}$ is a weighting factor.

Calculating the optimal deployment method using ILP is time-consuming. The problem becomes more difficult as the network size becomes more complex and the number of requested network services increases. To address this issue, the proposed system uses a machine learning model to determine the optimal placement method.

According to the precise definition of VNF deployment, the function evaluates the placement results in a one-time batch without continuous feedback through interaction

with the environment. In other words, a single optimal deployment solution can be determined and evaluated. Utilizing ILP as the ground truth, the proposed system learns the deployment using supervised learning. Figure 3.3 illustrates the overall learning system.

Algorithm 1 is the training process for the deployment function. A twin network, which replicates the current NFVI information, is first created during training. The requested network service is defined as an SFCR (Service Function Chain Request). Random objects (e.g., traffic flow, VNF, SFC) are generated and added to the network to introduce variations to the twin network. Then, the network and SFCR are input into the ILP to create optimal VNF placement. These labeled data form a 3-dimensional matrix of node numbers, VNF types, and VNF quantities.

The data generation process is repeated until sufficient data is produced. The generated data is then used as input for machine learning. The network data is represented in a graph format, consisting of a node matrix, adjacency matrix, and feature matrix, while the SFCR data is input as a single vector. Once the training is complete, the model is saved along with the network and SFCR information.

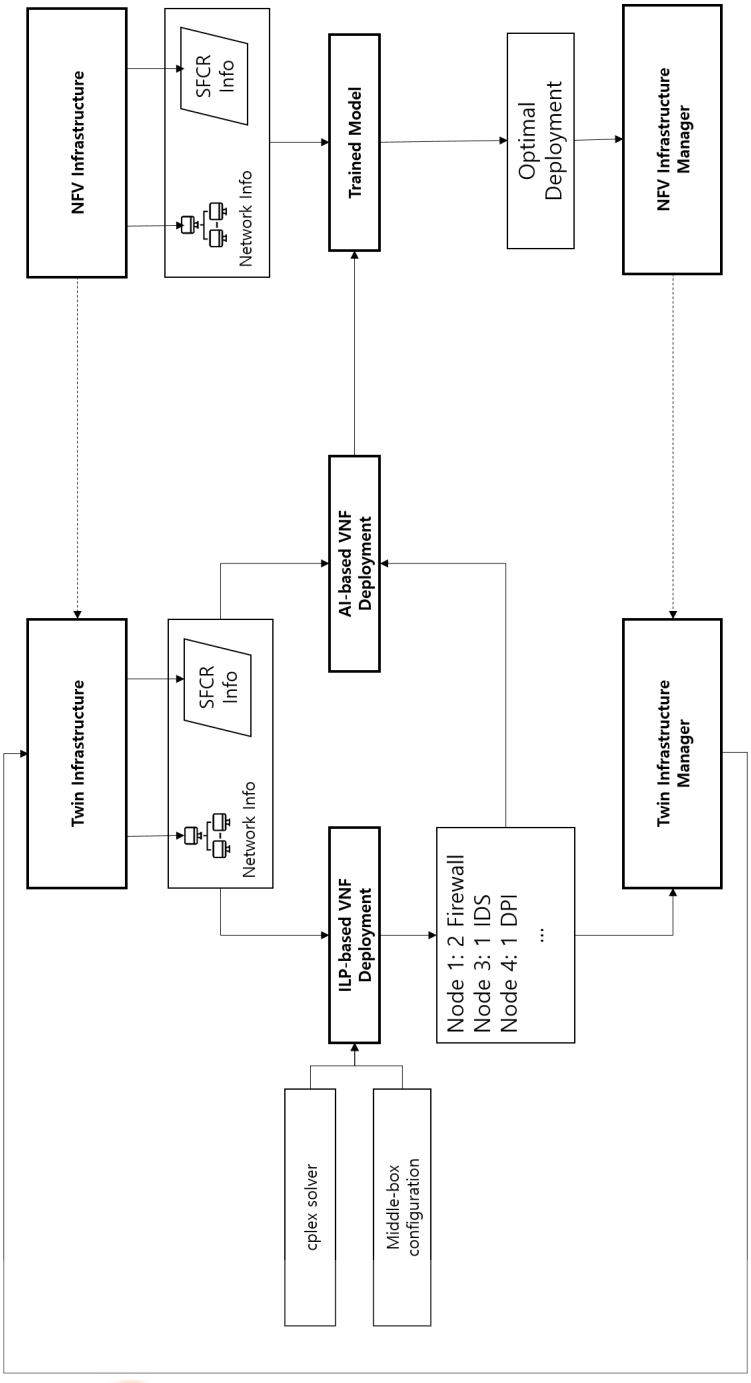

Figure 3.3: VNF Deployment by the proposed AI-NFV MANO

Algorithm 1 Training Process for Deployment Function

Input: Current NFVI information, SFCR information
Parameter: Number of dataset $\bar{D}$, Epoch $\bar{K}$
Output: Trained model with network, SFCR information
Create a twin network by replicating the current NFVI information
Configure ILP parameter for physical network information
while Before dataset $\bar{D}$ is gathered **do**
    Initialized twin network with input information
    Deploy random objects (VNFs, SFCs, SFCRs) to twin networks
    Preprocess the network and SFCR data for ILP
    Get an optimal solution for input data
    Store input and solution data
end while
Random Initialize learning parameters such as weights and biases
Preprocess the stored data for ML
while Before Epoch increased as $\bar{K}$ **do**
    Apply GNN to network data to obtain graph embedding
    Apply FNN to SFCR data to obtain an embedding
    Concatenate embedding data
    Apply FNN to get optimal deployment
    Optimize learning parameters
end while
Generate representation of input network.

3.4.2 Auto-scaling

The auto-scaling function optimally scales VNF instances according to the change in network traffic. The quality of service and the efficiency of network resources depend on when and how the scaling is performed. The operation of auto-scaling can be described as follows: First, when a specific SFC becomes the target for scaling, it is necessary to choose the scaling action among scale-out, scale-in, or doing nothing. If the action is selected as scale out, the next step is to decide which VNF to scale. This VNF selection is made on a tier basis. A tier is a unit that constitutes the chain of the SFC, where each tier is a set of VNFs of the same type. Figure 3.4 provides an example of a tier. After selecting the tier, the system prepares to install a new VNF and chooses the physical node where it will be installed. On the other hand, if the action is selected as scale-in, a tier is selected, and then the VNF to be removed is chosen.

The optimal scale-out policy involves selecting a tier with many overloaded VNFs and adding additional VNFs to the tier. During this process, the physical node for the new VNF should be chosen to minimize latency due to the SFC path. Similarly, the scale-in policy involves selecting a tier where resources are mostly wasted and removing VNFs from the SFC, choosing the VNF from the node causing the highest latency.

The optimal auto-scaling policy can be determined through interaction with the real environment. Due to the nature of SFCs, a series of connected VNFs affect each other. For example, if the VNFs in the earlier tiers of an SFC are overloaded, the system can detect the overload of those VNFs, but the VNFs in the later tiers may show a low load because the traffic does not reach them properly. However, since the problem is only observed in the earlier tiers, scaling is only performed on early tiers. Once the issue in the earlier tiers is resolved, the later tiers may become overloaded depending on the resource allocation. Also, some actions can introduce additional issues that were originally problem-free. For instance, if both the first and second tiers require scale-out, the node location of the scaled VNF in the first tier may be

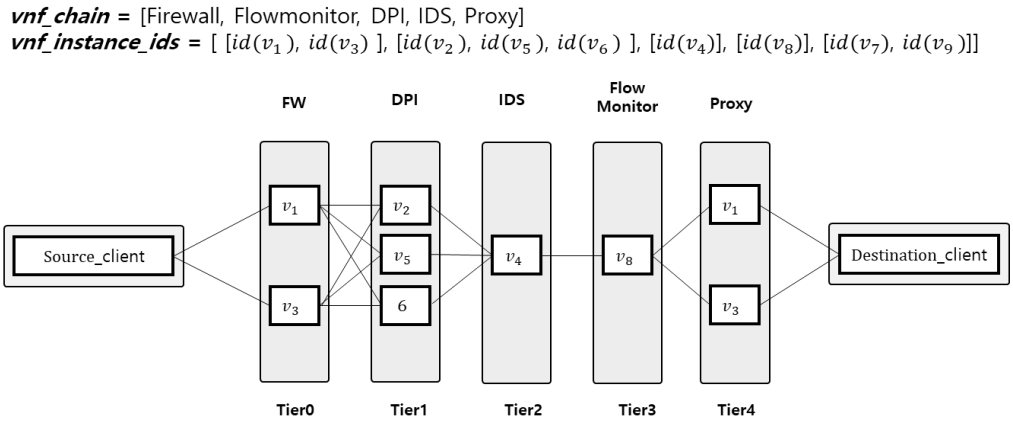

Figure 3.4: Example of Tier from the Service Function Chain

too far from the node location of the scaled VNF in the second tier, causing increased latency. Due to these characteristics of SFCs, auto-scaling must work continuously, and the model should design all possible scenarios to create optimal policies. However, implementing such a model is challenging, so this thesis uses reinforcement learning to generate scaling policies.

In this approach, scaling is performed on an SFC basis according to a precise definition of auto-scaling. When there are many SFCs in the network simultaneously, auto-scaling is performed independently for each SFC. The proposed system uses Deep Q-Network (DQN). Figure 3.5 illustrates the auto-scaling training process.

The state, action, and reward of auto-scaling is defined as follows:

- State: The state consists of the average CPU usage per tier, average memory usage per tier, average disk usage per tier, dropped packets/total packets, and distance score. Since the model focuses solely on the precise definition of auto-scaling, it does not consider the entire network information; instead, it only collects data from the target SFC. The distance score is the total length taken to reach the destination node from the source node through the chain. Additionally,

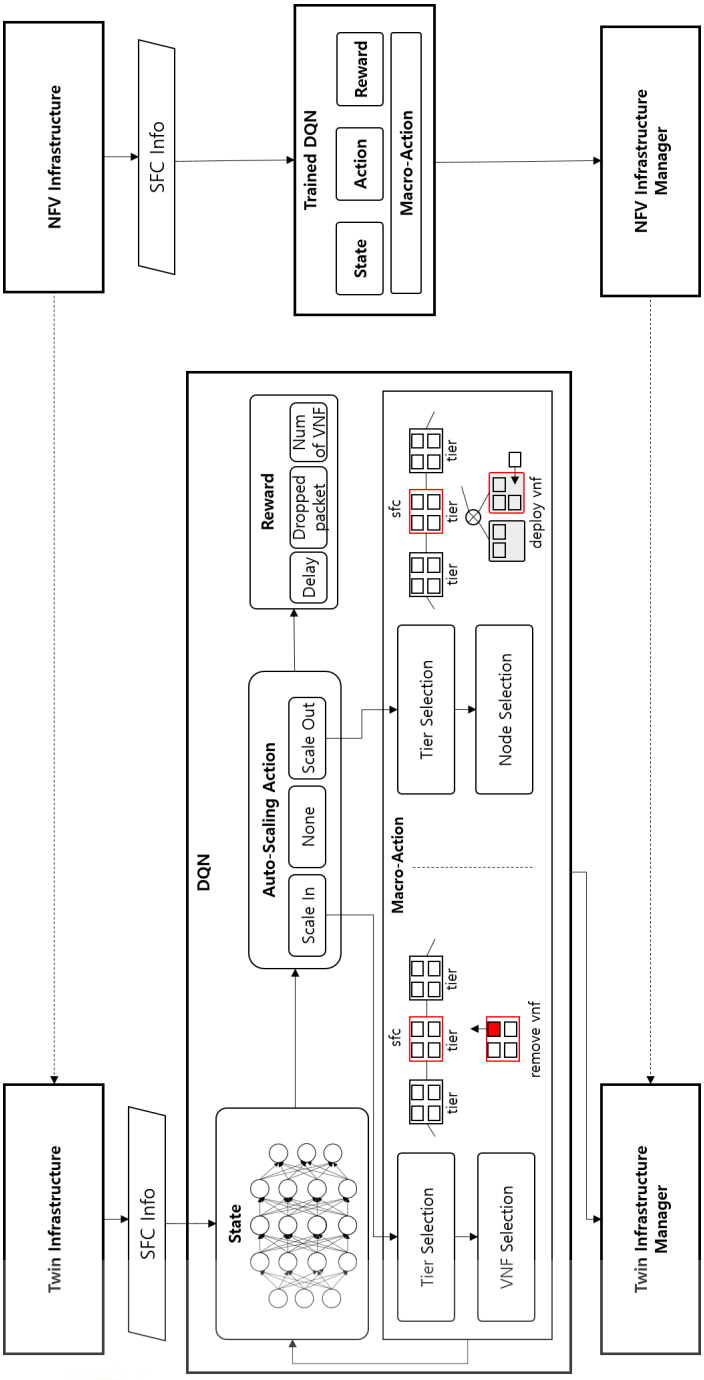

Figure 3.5: VNF Auto-scaling proposed by AI-NFV MANO

if the VNFs within a tier are deployed on different physical servers, the overall distance score tends to increase on average.

- Action: The action is defined as a set of scaling policies as scale-in, scale-out, or doing nothing. The actual scaling policy not only decides whether to scale but also requires selecting the tier, choosing the VNF to remove, and picking the node for installation. However, modeling all these actions requires a hierarchical RL model, which has multi-stage actions. However, by leveraging domain knowledge of the network, the system can avoid complicated actions. The proposed RL uses the macro-actions method to decide only the scaling (in, out, or nothing), and the subsequent actions (tier selection, VNF selection, node selection) are determined sequentially and optimally. Once scaling is decided, the subsequent actions are considered predetermined, and calculated optimally. The definition of macro-actions is defined as follows:

  - scale-out: Macro-action ensures that the number of changed instances falls within the maximum instance range, and masks out the tiers that exceed the allowable range. It calculates the expected changed resource utilization values (Equation 3.5) using CPU and memory resources. The distance score calculates the distance (number of hops) from the preceding and succeeding tiers' VNFs or the source and destination nodes for each tier's VNFs. Therefore, in scale-out, the cost function (Equation 3.6) selects the tier with higher resource utilization, and if the resource utilization is the same, it chooses the tiers that cause slightly more delay. After selecting a tier, the node is selected by using the distance score.

$$\text{Resource utilization} = \sum_{\bar{t} \subseteq vii(c)} \sum_{v \in \bar{t}} u_1(cur\_cpu(v)) + u_2(cur\_ram(v)) \tag{3.5}$$

$$\text{Cost function} = e^{\text{resource utilization}} + \text{distance score} \tag{3.6}$$

– scale-in: Macro-action ensures that the number of installed instances falls within the minimum range, and masks out the tiers that exceed the allowable range. Then, construct the cost function (Equation 3.7) to select the tier with lower resource utilization. After selecting a tier, the node is selected by using the expected distance score.

$$\text{Cost function} = e^{-\text{resource utilization}} \tag{3.7}$$

- Reward: The reward is defined as Equation 3.8. The equation minimizes the sum of delay, packet loss, and the total number of instances. Specifically, the delay is minimized along the SFC path, and packet loss is minimized to prevent issues caused by resource overload. The number of instances is minimized to avoid excessive waste of network resources.

$$\begin{aligned} \text{Reward} = -(&\alpha e^{(1+meausred\_delay)} \\ &+ \sum_{\bar{t} \subseteq vii(c)} \sum_{v,u \in \bar{t}} \beta \log(1 + cur\_dropped\_tx, rx(e_{v,u})) \\ &+ \gamma \log \left(1 + \sum_{\bar{t} \subseteq vii(c)} \sum_{v \in \bar{t}} 1 \right)) \end{aligned} \tag{3.8}$$

Algorithm 2 explains the training process for the auto-scaling function. During training, a twin network is created by replicating the current NFVI information, and a target SFC is selected. Random objects are generated and deployed to the network. In DQN training, the exploration value decreases, and optimal actions are selected instead of random ones. Minimum and maximum values for VNFs per tier are specified to ensure actions do not exceed these limits, and each action triggers the corresponding macro-action automatically.

Algorithm 2 Training Process for Auto-scaling Function

Input: Current NFVI information, SFCR information
Parameter: Memory capacity N, Update interval step, E, Epoch $\bar{K}$
Output: SFC information
Create $\bar{A}$ twin networks by replicating the current NFVI information
Deploy random objects (VNFs, SFCs, SFCRs) to twin networks
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\hat{\theta}$
while before θ convergence **do**
    Initialize and preprocess sequence s_1
    for each epoch **do**
        With probability ϵ select a random action a_t
        Otherwise select $a_t = \arg\max_a Q(\phi(s_t), a; \theta)$
        if a is scale-out **then**
            Select tier $\bar{t} = \arg\max_{\bar{t}} e^{\text{resource utilization}} + \text{distance score}$
            Select node $n = \arg\min_n$ distance score
            Install VNF to node n
        else if a is scale-in **then**
            Select tier $\bar{t} = \arg\max_{\bar{t}} e^{-\text{resource utilization}}$
            Select VNF $v = \arg\max_v$ distance score
            Remove VNF v
        end if
        Set $s_{t+1} = s_t, a_t, r_t, s_{t+1}$
        Store transition $(\phi(s_t), a_t, r_t, \phi(s_{t+1}))$ in D
        Sample random mini-batch of transitions $(\phi(s_j), a_j, r_j, \phi(s_{j+1}))$ from D
        Optimize scaling action wrt θ
        Every E steps update the target action value function $\hat{Q}$
        Execute action a_t
    end for
end while
Generate representation of input network.

3.4.3 Service Function Chaining

Service function chaining improves service quality by changing the SFC chain and the quality depends on how the chain is altered. This function operates similarly to auto-scaling, but it performs the precise scope of chaining without considering the creation or deletion of additional VNFs. The operation of service function chaining can be described as follows: First, when a specific SFC becomes the target for chaining, it is necessary to decide whether to change the chain or not. Then, the tier of the SFC to be changed is selected, and the VNF to be removed from the SFC is chosen. After that, a model chooses VNF to be added to the SFC. The optimal chaining policy is to remove abnormal VNFs from the SFC and add normal VNFs to the SFC.

The optimal policy for service function chaining, like auto-scaling, can be determined through interaction with the real environment. To create an optimal chaining policy, it is necessary to model all possible scenarios that may occur, similar to auto-scaling, due to the characteristics of SFCs. However, creating such a model is also challenging, therefore reinforcement learning is used to generate chaining policies. When there are many SFCs in the network simultaneously, chaining is performed independently for each SFC. The proposed system uses the DQN model to learn optimal service function chaining policy, Figure 3.6 illustrates the training process.

The state, action, and reward of service function chaining is defined as follows:

- State: The state consists of the average CPU usage per tier, average memory usage per tier, average disk usage per tier, dropped packets/total packets, and distance score as auto-scaling.

- Action: Actions determine whether to proceed with chaining and apply macro-actions like auto-scaling. The detailed operation of macro-actions includes selecting the tier to be changed, choosing the VNF to be removed from the SFC, and selecting the VNF to be included in the SFC. Tier selection and VNF selection are based on the current state information. Macro-action masks and selects tier ensuring there is at least one VNF exists. It calculates the current resource

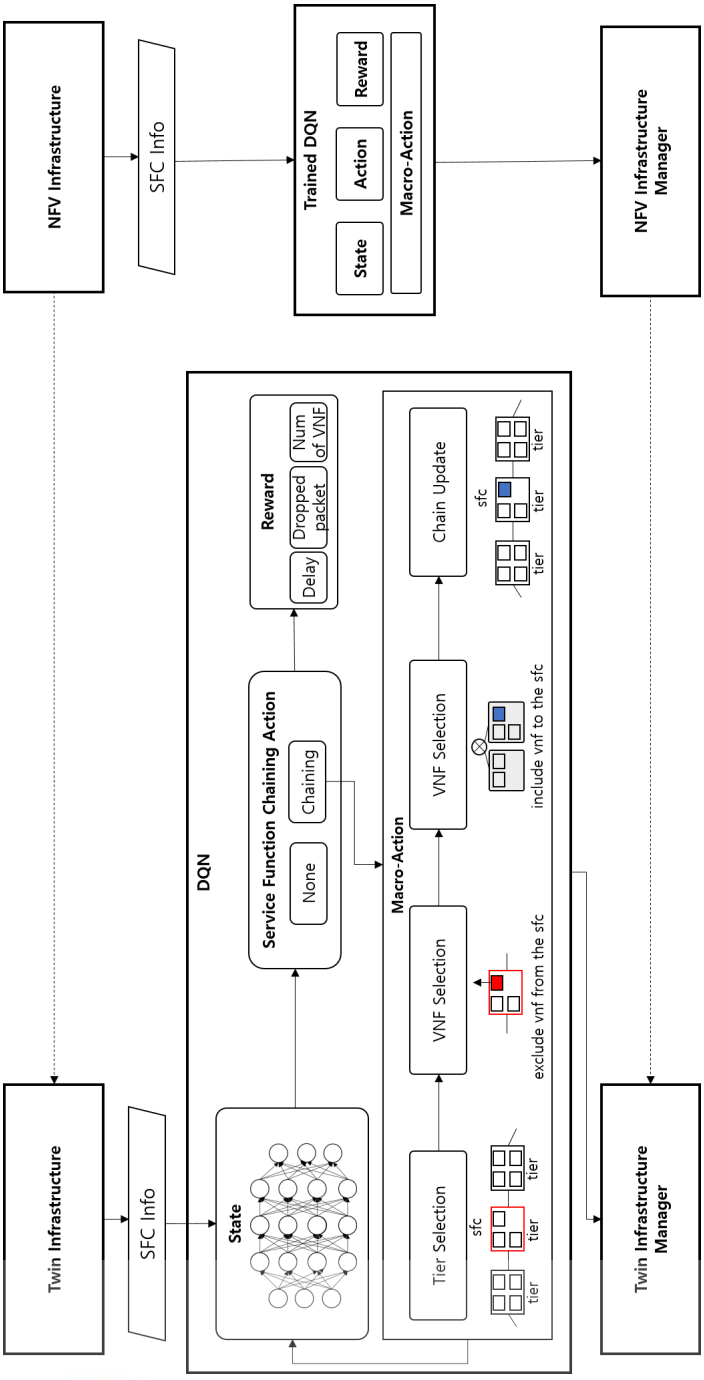

Figure 3.6: Service Function Chaining proposed by AI-NFV MANO

utilization values (Equation 3.5) and distance score as same as auto-scaling. The cost function is the same as the auto-scaling cost function (Equation 3.6).

- Reward: The reward function is the same as Equation 3.8.

Algorithm 3 explains the training process for the service function chaining. During training, a target SFC is selected and a twin network is created by replicating the current NFVI information. Random objects are deployed to the network, and some stress is injected into several VNFs.

Algorithm 3 Training Process for Service Function Chaining Function

Input: Current NFVI information, SFCR information
Parameter: Memory capacity N, Update interval episode, E, Epoch $\bar{K}$
Output: SFC information
Create $\bar{A}$ twin networks by replicating the current NFVI information
Deploy random objects (VNFs, SFCs, SFCRs) to twin networks
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\hat{\theta}$
while before θ convergence **do**
    Initialize and preprocess sequence s_1
    for each epoch **do**
        With probability ϵ select a random action a_t
        Otherwise select $a_t = \arg\max_a Q(\phi(s_t), a; \theta)$
        if a is chaining **then**
            Select tier $\bar{t} = \arg\max_{\bar{t}} e^{\text{resource utilization}} + \text{distance score}$
            Select VNF $v_1 = \arg\max_v e^{\text{resource utilization}} + \text{distance score}$
            Select VNF $v_2 = \arg\min_v e^{\text{resource utilization}} + \text{distance score}$
            Exclude v_1 and include v_2 to SFC
        end if
        Set $s_{t+1} = s_t, a_t, r_t, s_{t+1}$
        Store transition $(\phi(s_t), a_t, r_t, \phi(s_{t+1}))$ in D
        Sample random mini-batch of transitions $(\phi(s_j), a_j, r_j, \phi(s_{j+1}))$ from D
        Optimize scaling action wrt θ
        Every E steps update the target action value function $\hat{Q}$
        Execute action a_t
    end for
end while
Generate representation of input network.

3.4.4 Consolidation

The consolidation function aims to maintain the quality of network services while minimizing resource fragmentation costs. In an NFV environment, the consolidation function achieves resource and power efficiency by concentrating distributed VNFs on nodes. The service quality and resource efficiency depend on which VNFs are selected and to which nodes they are consolidated. The detailed operation of consolidation is as follows: First, node selection entails identifying the suitable server from multiple servers housing VNFs. Subsequently, VNF selection involves choosing which VNFs within the selected server will undergo migration. Lastly, VNF migration determines the physical server to which the selected VNFs will be relocated. Reducing the routing path of SFC's VNF instances can mitigate latency.

The effectiveness of consolidation policies can be evaluated through interaction with the actual environment. Given the nature of consolidation, which entails consolidating VNFs on specific nodes and managing multiple resource allocations, additional sequential actions are necessary for optimization, illustrated in Figure 3.7. For example, Node 1 may have surplus CPU capacity but insufficient memory, whereas Node 2 may have surplus memory but insufficient CPU capacity. By reallocating specific VNFs through Node 3, both nodes can accommodate two additional VNFs currently hosted on Node 3. These processes pose challenges for solutions based on rule-based approaches or dynamic programming that consider every possible scenario. Therefore, this system employs reinforcement learning to derive optimal policies.

To simplify the consolidation process, the system integrates the node selection process into the VNF selection process, solving the consolidation problem in two stages. Each stage is learned using different neural networks but with identical training cycles. Due to the structure of the learning model, achieving adequate exploration with a conventional value-based learning approach is challenging. Thus, the system adopts a policy-based approach utilizing the PPO model for training. Figure 3.8 illustrates the training structure.

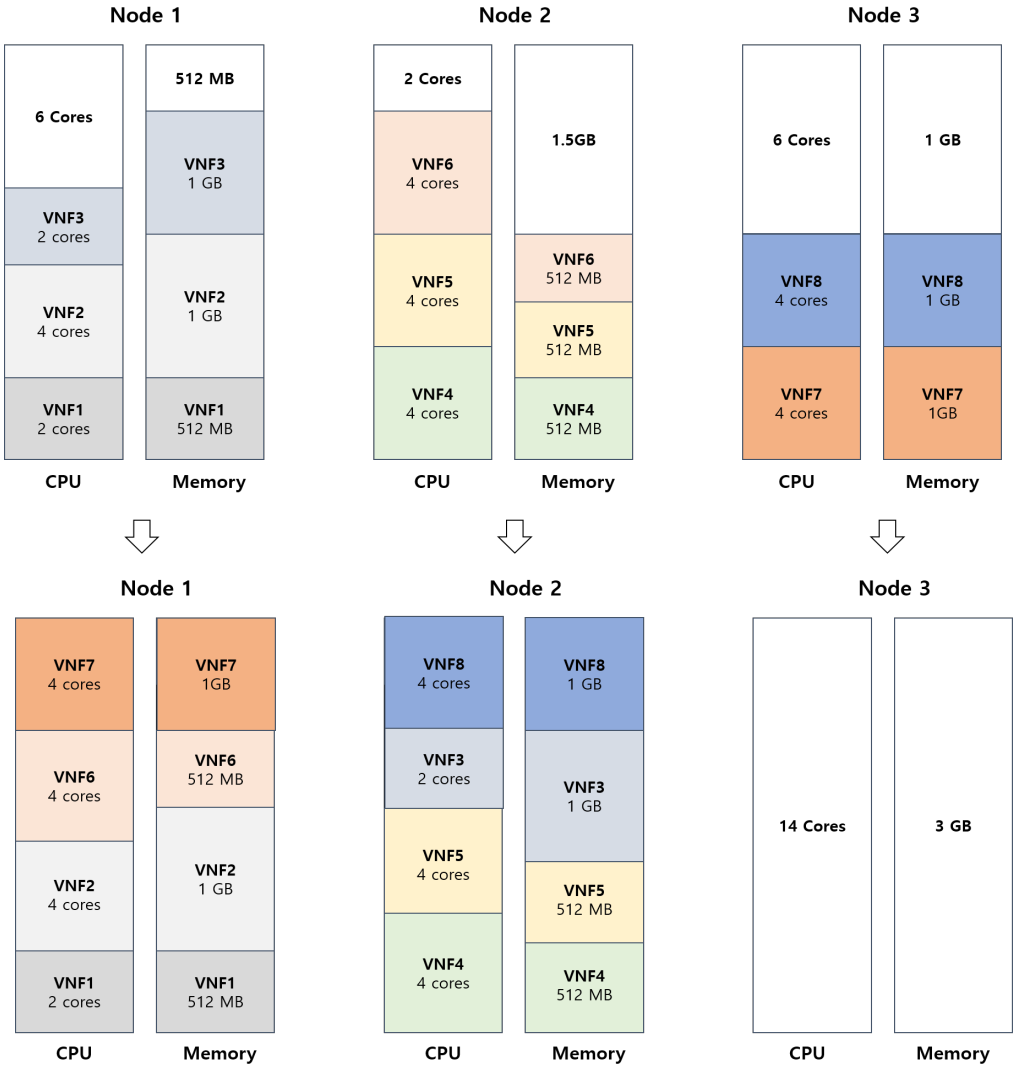

Figure 3.7: Example of VNF Consolidation Process

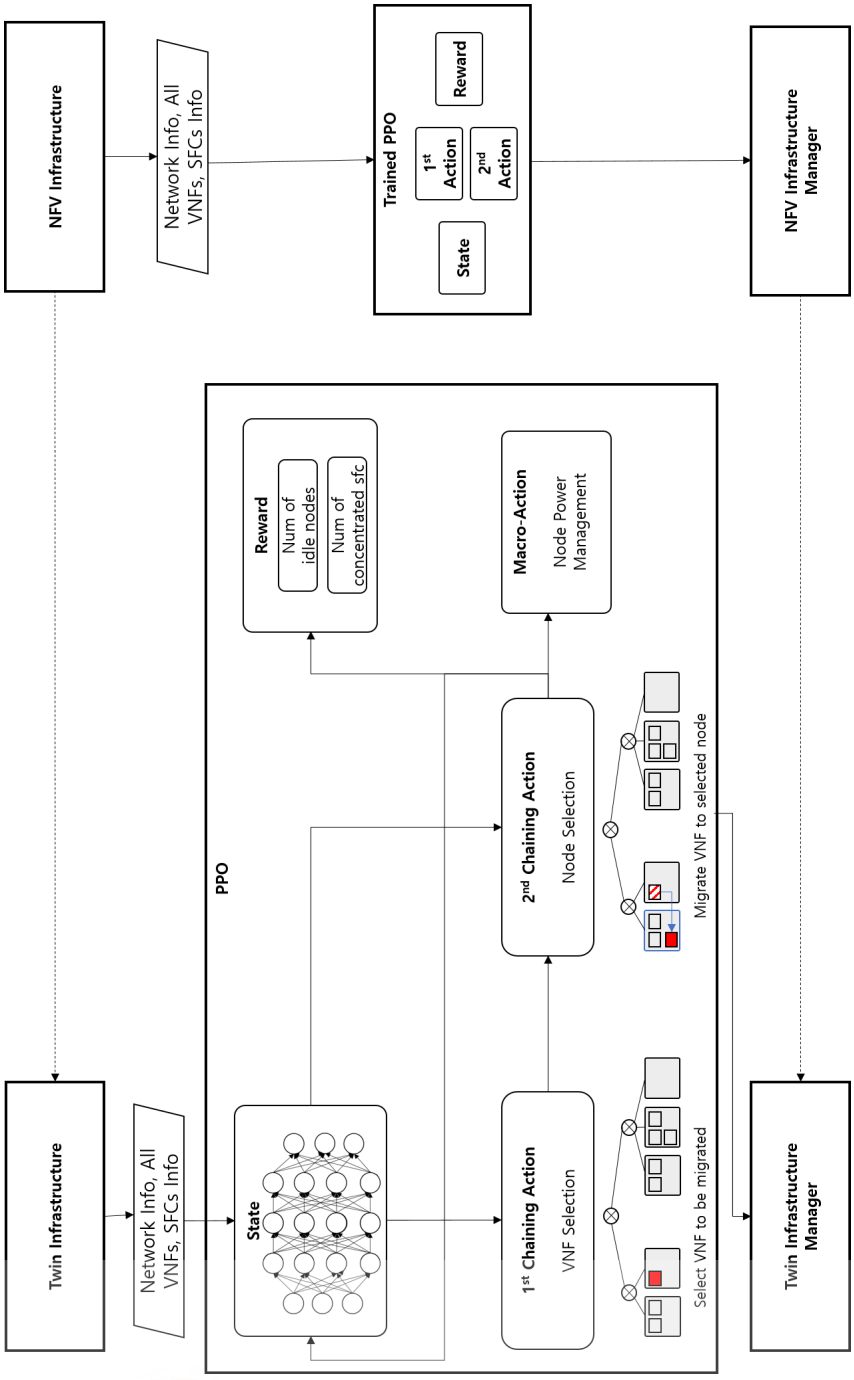

Figure 3.8: VNF Consolidation proposed by AI-NFV MANO

The state, action, and reward of consolidation are defined as follows:

- State: The state comprises individual data for all nodes, VNFs, and SFCs. Nodes provide details regarding CPU, memory, and disk capacities, while VNFs include resource requirements for flavors and their association with specific SFCs.
- Action: Actions are executed in two stages. The first stage involves VNF selection, and the second stage involves node selection. Unlike auto-scaling and service function chaining, the complexity of consolidation makes the use of macro-actions challenging. Nevertheless, it can employ additional actions (such as power management) as macro-actions.
- Reward: The system's reward is defined as shown in Equation 3.9. The total number of servers is denoted as $n(N)$, where $n(N^0)$ represents the number of servers without any installed VNFs. The total number of SFCs is denoted as $n(C)$, and $n(C^\forall)$ indicates the number of SFCs with all VNFs installed on the same node.

$$\text{Reward} = \frac{n(N^0)}{n(N)} + \frac{n(C^\forall)}{n(C)} \tag{3.9}$$

Algorithm 4 explains the training process for the consolidation function. During training, it proceeds through the stages of VNF selection and VNF migration. The default maximum episode length is set to twice the number of VNFs, but it can vary depending on the complexity of the problem. Upon reaching the maximum episode length, the episode concludes, and the network is reset to its initial state for subsequent training.

Algorithm 4 Training Process for VNF Consolidation Function

Input: Current NFVI information
Parameter: Agent $\bar{A}$, Maximum step size $\bar{P}$, Epoch $\bar{K}$
Output: Trained model with network information
Create $n(\bar{A})$ twin networks by replicating the current NFVI information
Deploy random objects (VNFs, SFCs, SFCRs) to twin networks
Random Initialize θ^{vnf} and θ^{node}
$\theta^{vnf}_{\text{old}} \leftarrow \theta^{vnf}$, $\theta^{node}_{\text{old}} \leftarrow \theta^{node}$
while before θ convergence **do**
    for $\bar{A} = 1, 2, \cdots, N$ **do**
        Run VNF Selection policy $\pi_{\theta^{vnf}_{\text{old}}}$ in twin network for $\bar{P}$ steps
        Run Node Selection policy $\pi_{\theta^{node}_{\text{old}}}$ in twin network for $\bar{P}$ steps
        Compute VNF Selection estimating advantage advantage $A_1, A_2, \cdots, A_T$
        Compute Node Selection estimating advantage advantage $A_1, A_2, \cdots, A_T$
    end for
    Compute clipped surrogate objective with with $\bar{K}$ epochs
    Optimize VNF Selection wrt θ^{vnf} with $\bar{K}$ epochs
    Optimize Node Selection wrt θ^{node} with $\bar{K}$ epochs $\theta^{vnf}_{\text{old}} \leftarrow \theta^{vnf}$, $\theta^{node}_{\text{old}} \leftarrow \theta^{node}$
end while
Generate representation of input network.

3.5 Learner Controller

To use machine learning models for network management, it is necessary to ensure that the learning models provide optimal network management policies in response to network conditions. However, in practice, creating optimal policies in response to all network changes using machine learning models poses significant challenges, such as requiring a vast amount of data and extensive training time. The proposed system acknowledges these issues and modifies the strategy to provide machine learning models from a different perspective. Given the complexity and dynamic nature of networks, the learning problem is challenging, rendering it impractical to generate optimal policies. Thus, the system mitigates the problem's complexity and enhances learning performance by focusing machine learning efforts on specific network contexts.

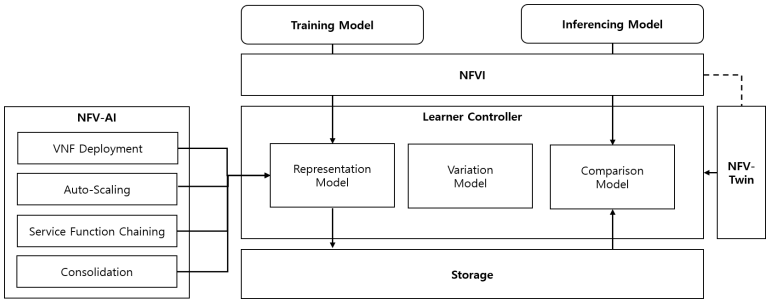

Figure 3.9: AI-Leaner Controller proposed by AI-NFV MANO

Instead of creating machine learning models that generate correct answers for all scenarios, the system generates policies only for targeted network states. For instance, the system may skip learning for rare network states and concentrate exclusively on frequent network states. However, this learning approach ensures performance only for certain network states and does not guarantee performance during dynamic changes. To address this issue, the system defines various network states that can occur in the short term based on the current network state and generates machine-learning models for those networks. The trained models are stored along with the training environ-

ment information and later used for inference based on the environment information. Through this process, the system can use optimized learning models for various network environments without additional training time. Figure 3.9 illustrates the proposed Learner Controller. The controller takes the current network environment as input through the NFVI and creates a twin network using the NFV-Twin. The generated twin network is modified in various ways according to the variation model. After completing the training of the NFV-AI model, the training environment is stored using the representation model. Subsequently, when a network management policy is required, the comparison model is employed to deduce the model trained in the environment most similar to the current network state, thereby generating the policy. The representation model is established using input data from each NFV-AI management model, employing cosine similarity for the comparison model. The entire system operates as follows:

- Define the primary network environments and permanently store and train them.
- Periodically train using the current network environment.
- Introduce variations to each environment and perform additional training.
- Train on randomly selected network environments.
- Remove infrequently used training models.

3.6 Policy Controller

Each NFV management policy must avoid conflicts and achieve overall network optimization. Therefore, this thesis prioritizes NFV management functions to avoid conflicts. When analyzing the characteristics of management functions, VNF deployment is crucial for initiating services, while other functions manage the already deployed ones. To ensure the smooth operation of new services, this thesis regards the deployment function as the highest priority. Auto-scaling and service function

chaining share many similarities, as both aim to enhance network quality through chaining modifications. However, auto-scaling involves deployment and time costs by installing new VNFs, whereas service function chaining is less costly since it selects already deployed VNFs. Thus, service function chaining should take precedence over auto-scaling. The consolidation function aggregates network functions on servers. Conducting consolidation operations during network congestion can degrade service quality, thus consolidation should be executed during periods of low traffic. Also, the consolidation function may require frequent migrations. During such operations, it must ensure that actions taken by other management functions do not rollback or cause oscillations.

The overall flow of management policies is illustrated in Figure 3.10. In each time window W, the VNF deployment module checks and executes requests for new services, followed by the execution of all required service function chaining actions for each SFC. Unlike service function chaining, auto-scaling does not perform all actions at once and can be interrupted by deployment or service function chaining during its operations. If another management function is executed during an auto-scaling operation, the current auto-scaling policy is discarded and restarted from the beginning. The consolidation function operates by temporarily pausing the execution of other functions, except for the deployment function.

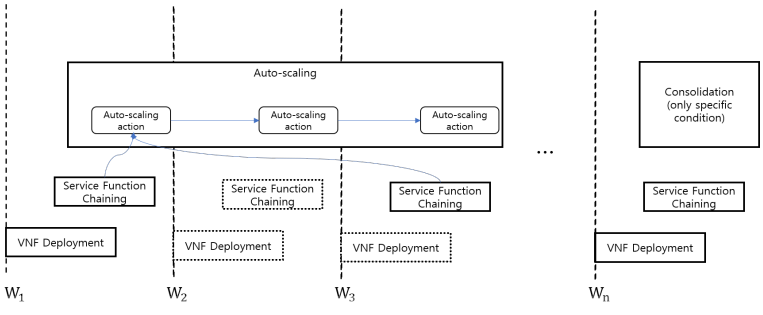

Figure 3.10: Policy flows of proposed AI-NFV MANO

IV. Implementation

4.1 Testbed

The proposed NFV-MANO was implemented using OpenStack, depicted in Figure 4.1. The testbed employs a multi-node architecture, primarily comprising a controller node and compute nodes. The controller node hosts most OpenStack services, while compute nodes are dedicated to installing and operating VNF instances. Various agents and schedulers operate as daemons on the controller node to manage core OpenStack functions. OpenStack's Heat serves as the resource orchestration engine, and Tacker acts as the NFVO/VNFM, facilitating basic NFV environment orchestration. Compute nodes utilize the KVM hypervisor to execute VNFs, each integrating an Open vSwitch software switch to handle VM network connections.

NFV-MON, deployed on the monitoring node, gathers data from the testbed, while NFV-AI on the AI node generates policies for NFV-MANO. Detailed specifications of the compute nodes are outlined in Table 4.2. A DPDK-based delay emulator named DEMU simulates the MEC environment. Edge1 and Edge2 experience an 8ms round-trip delay between nodes and the OpenFlow switch, while the Data Center experiences a 30ms delay with the switch. Edge3 does not experience additional delay from DEMU.

Table 4.2 displays the resource requirements for each flavor, where each flavor is associated with a specific image. These images include software components essential for network functionality. Administrators have the capability to create new flavors and images tailored to network demands, as well as to capture snapshots of currently active VNFs.

In OpenStack, VNFs can access external networks via port forwarding. To streamline management, a VNF named Connector has been deployed for port forwarding,

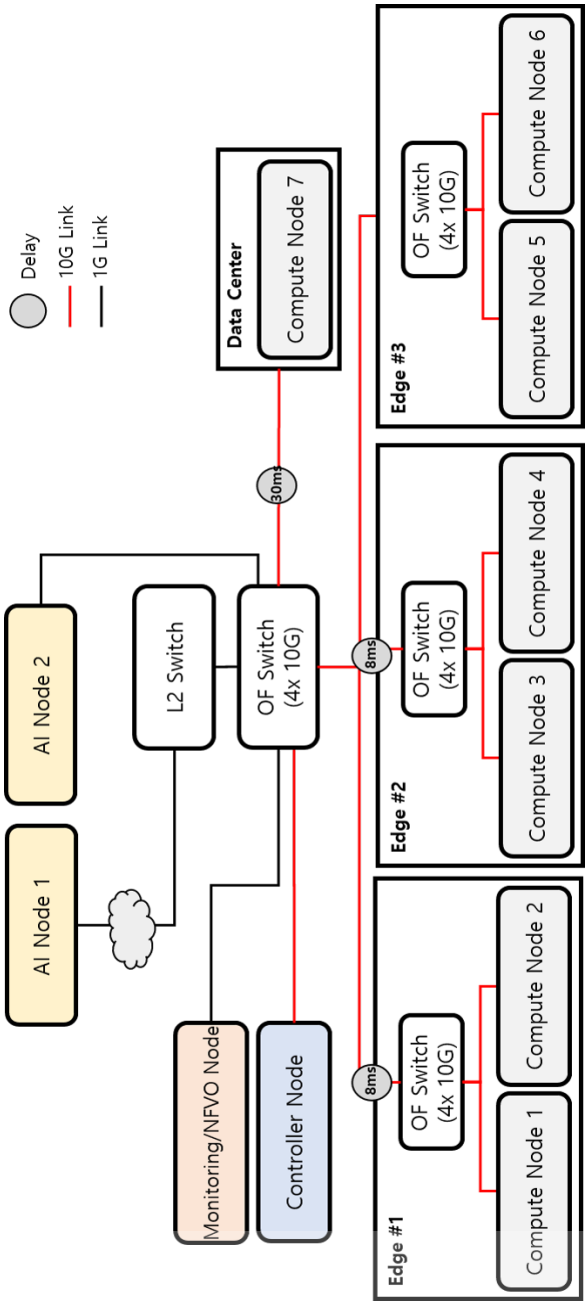

Figure 4.1: Physical Testbed

enabling direct external network control over VNFs. The VIMI interfaces with this VNF, providing functionalities such as traffic generation via iperf3, delay measurement using Apache Benchmark, and stress testing with stress-ng.

NSs operate at the SFC level. The proposed system leverages OpenStack's networking-sfc[1] and Tacker for SFC management. With networking-sfc, VM ports can be sequentially interconnected. Tacker compiles predefined descriptors using Heat, allocates resources within the testbed, and orchestrates actions through OpenStack's nova[2] and neutron[3] service. NFV-AI generates policies for the updating and optimization of SFCs, ensuring efficient management.

[1] https://docs.openstack.org/networking-sfc/
[2] https://docs.openstack.org/nova
[3] https://docs.openstack.org/neutron

	vCPU [Cores]	RAM [GB]	Disk [TB]
Compute Node 1	32	32	0.5
Compute Node2	32	32	0.5
Compute Node3	24	24	2
Compute Node4	24	24	1
Compute Node5	16	24	1
Compute Node6	6	12	1
Compute Node7	32	96	2
Total	166	244	8

Table 4.1: Environment Specification

	vCPU [Cores]	RAM [GB]	Disk [GB]	Software
Default	32	32	0.5	ubuntu
Firewall	32	32	0.5	iptable
Flowmonitor	24	24	2	ntop
DPI	24	24	1	nDPI
IDS	16	24	1	suricata
Proxy	6	12	1	apach

Table 4.2: VNF Flavor Specification

4.2 NFV-MON

NFV-MON gathers resource usage and log data from VNFs and physical servers within the NFV environment. It utilizes the open-source monitoring tool Collectd[4], for data collection, preprocessing, and storage in the Gnocchi[5] database, aggregating data into averages, minimums, maximums, etc. Integration includes Collectd-Web for real-time monitoring and Grafana[6] for querying stored data via Gnocchi's web dashboard.

The monitoring module also captures traffic statistics and saves packets from physical servers, VMs, and containers every 10 seconds in pcap format. These pcap files are then analyzed using the open-source traffic analysis tool CIC Flowmeter[7], storing statistical information on each traffic flow in a database and optionally exporting it to CSV format.

To facilitate seamless integration with NFV-AI, each core function is modularized via REST-API, and Apache Kafka[8] serves as both a database and message broker, constructing a robust pipeline for data storage and processing. Ad hoc requests are consolidated into single queries, while time-series data aggregation from diverse sources utilizes a pub/sub-messaging approach to minimize overhead.

Due to limitations with hypervisors, conventional hardware-based TAP devices cannot monitor VNFs effectively. As a workaround, virtual TAP (vTAP) functionality is employed to replicate packets exchanged between VNFs. Leveraging the vTAP ONOS application, packets are duplicated from the host virtual switch, Open vSwitch, to a monitoring node. Open vSwitch operates in both kernel mode (OVS) and DPDK mode (OVS-DPDK) to gauge the Rx throughput of the monitoring node, assessing the processing capacity of the duplicated packets.

[4] https://github.com/collectd/collectd
[5] https://github.com/gnocchixyz/gnocchi
[6] https://github.com/grafana/grafana
[7] https://github.com/ahlashkari/CICFlowMeter
[8] https://kafka.apache.org/

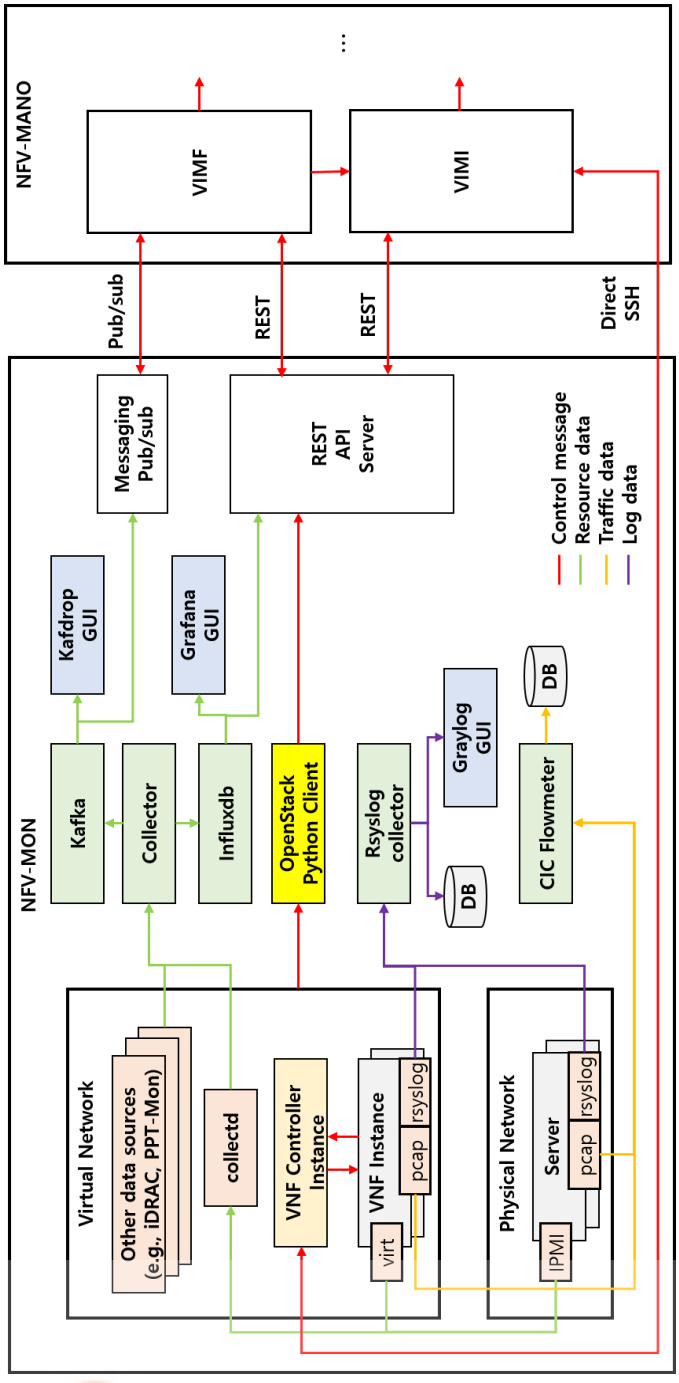

Figure 4.2: NFV-MON proposed by AI-NFV MANO

4.3 NFV-AI

Each NFV management function is provided as a modular service. Figure 4.3 illustrates the architecture of NFV-AI services. Each service's functionality is accessible via a REST API, enabling NFVI management through VIMF and VIMI. Depending on user needs, modules can register the management of diverse network elements as services, either manually or automatically. Registered elements are independently managed through dedicated threads. The learner controller module evaluates each element to determine if a trained model is available, proceeding with inference and training as necessary. Existing trained models are utilized for optimal policy inference, while absent models prompt training using a synthetic, modified twin network generated by the variation model. Upon completion of training, network environment details along with the trained model are stored using the representation model.

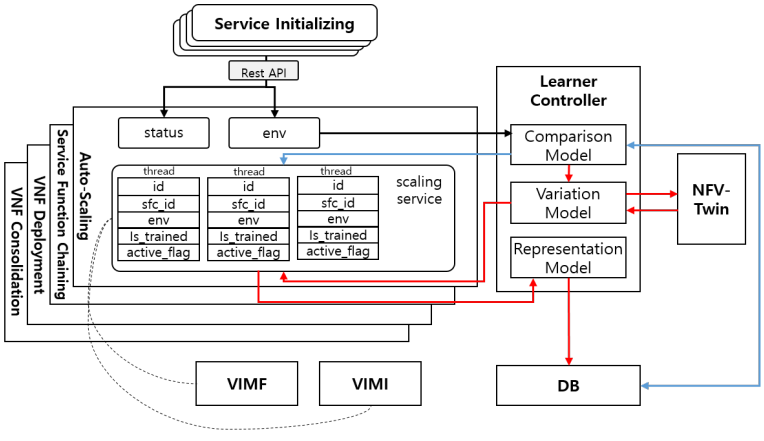

Figure 4.3: Architecture of Proposed AI-based NFV Management Service

4.3.1 Deployment

The deployment function serves as a module for providing network services. Administrators register the desired network configuration, with the entire network being registered by default. Upon registration, the module creates twin networks and gen-

erates random SFCRs to accommodate future network service requests. To simulate diverse network conditions, objects like VNFs, SFCs, and traffic can be generated and manipulated within the network based on learner controller. However, networks manipulated beyond a specified threshold are considered new networks and stored separately from the existing dataset.

Stored network datasets undergo linear operations using the CPLEX [44] program to produce optimal deployment labels, saved in a 3-dimensional matrix format (node number, VNF type, number of instances). Figure 4.4 depicts the neural network architecture for VNF deployment. During training, the neural network receives input from collected network information, SFCR information, and labeling data. Network information is transformed into node feature matrices, adjacency matrices, and edge feature matrices for graph-based representation.

During feature extraction, two graph neural network layers perform node embedding with 512 neurons each, while SFCR data undergoes embedding via fully connected (FC) layers with 128 neurons each. The resulting embeddings are flattened and concatenated. The concatenated data then passes through two FC layers, with 512 and 120 neurons respectively, followed by a 3D softmax layer to produce labeling values. These values are used to compute the loss using categorical cross-entropy. Table 4.3 details the training model parameters.

Learning rate	0.001
Optimizer	Adam
Size of batch	64
Update epochs	10,000

Table 4.3: Hyperparemter of VNF Deployment

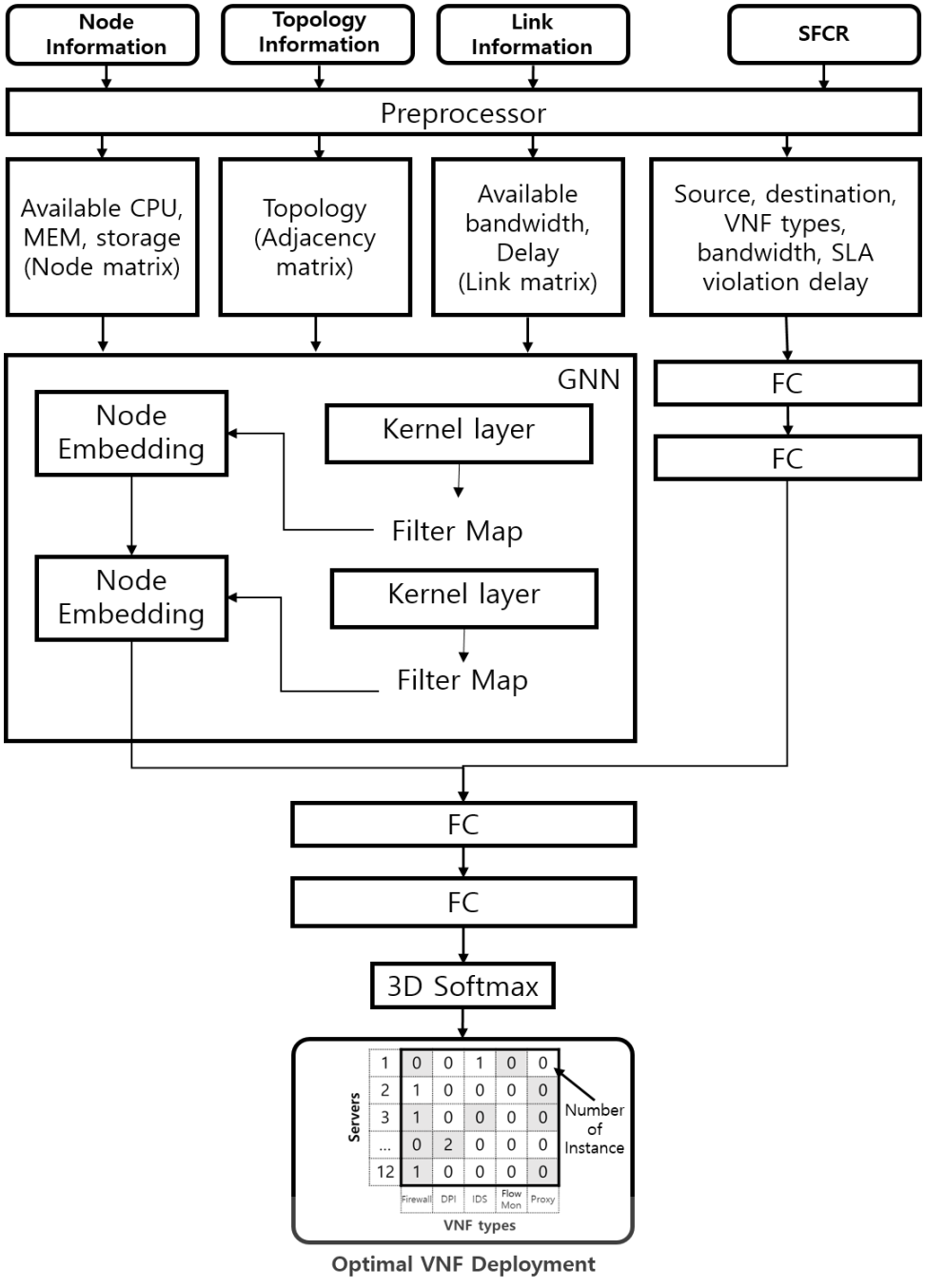

Figure 4.4: Neural Network Model of the VNF Deployment

4.3.2 Auto-scaling and Service Function Chaining

Auto-scaling registers the target SFCs for services, and these registered SFCs are managed independently. During the learning process, these registered SFCs scale the twin network and learn from the outcomes of their actions. Furthermore, the learner controller generates randomly modified SFCs and network environments for additional learning in various network scenarios.

Figure 4.5 depicts the RL model of VNF auto-scaling. In the learning process, the agent establishes a replay memory, a q-network, and a target q-network. Once sufficient data is accumulated in the replay memory, DQN training commences. The agent takes the state as input and learns through five FC layers, each comprising 128 neurons, to generate scaling actions. Upon generating a scaling action, the process dynamically selects tiers, VNFs, and nodes, leading to the deployment or removal of VNFs. Table 4.4 lists the hyperparameters of auto-scaling. The service function chaining shares the same structure and parameters as auto-scaling, differing only in the actions performed.

Learning rate	0.01
Optimizer	Adam
Discount factor	0.98
Initial epsilon	0.9
epsilon decay	0.001
Size of batch	16
Update epochs	50
Maximum number of step	4x(number of tier)
Maximum number of episode	1,000
Maximum size of memory	2,000

Table 4.4: Hyperparemter of auto-scaling and service function chaining

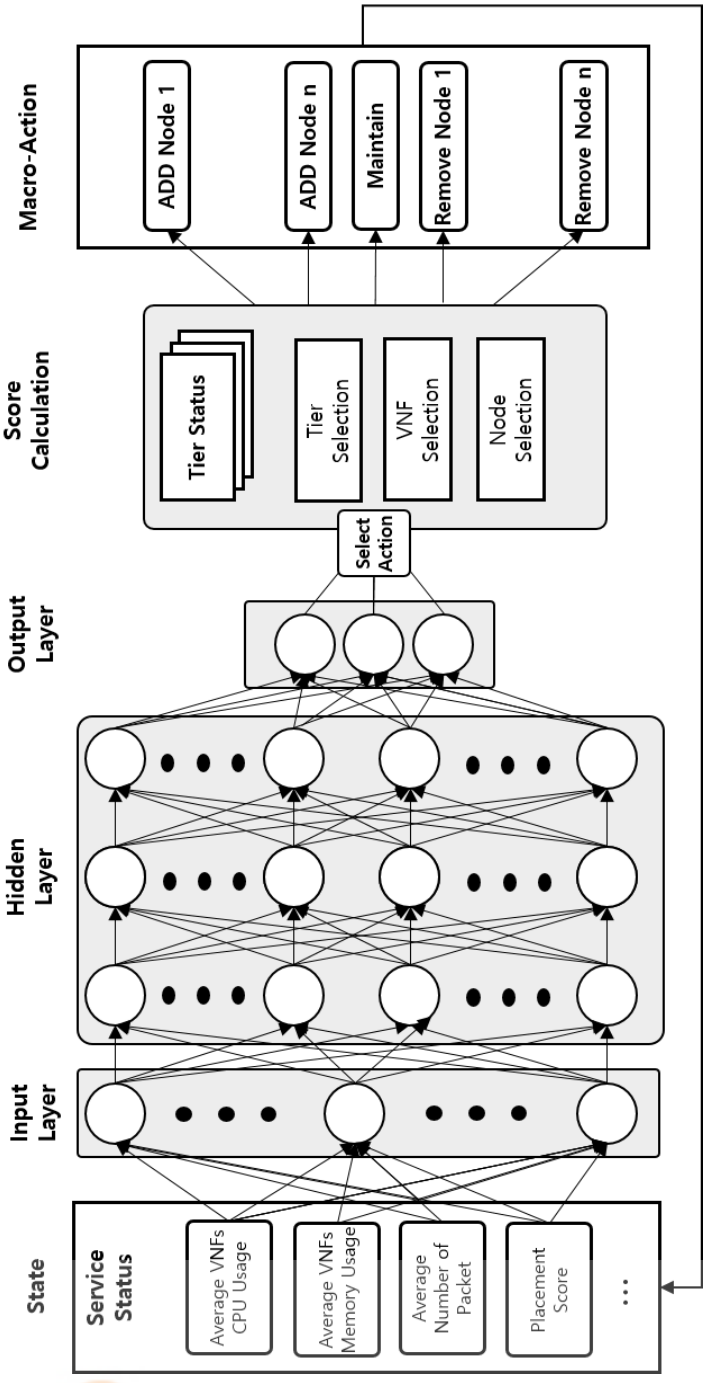

Figure 4.5: DNN Model of the Auto-scaling

4.3.3 Consolidation

The consolidation service can manage multiple networks. By default, it registers the entire network, and individual edge networks can also be registered to enable parallel management functions. Figure 4.6 depicts the RL model for VNF consolidation, comprising separate learning networks for VNF Selection and Node Selection.

For VNF selection, feature extraction is conducted using an FC layer with the RL state. Subsequently, the Self Attention Mechanism (SAM) is employed to estimate the hidden value associated with the current network state for each VNF. This iterative process involves two repetitions, with policy and value estimated using the critic network and actor network, respectively. Node selection utilizes actions from VNF Selection to filter the state as its input. Similarly to VNF selection, SAM estimates the hidden value associated with each node's VNF, while policy and value estimation leverages the critic network and actor network.

Table 4.5 outlines the parameter values of the consolidation. The policy and value learning rates for both VNF selection and node selection are set to 0.001, with policy gradient clipping at 0.1. Each policy has an entropy loss weight of 0.01 to encourage exploration. The discount factor is 0.99, and the GAE parameter is set to 0.97, facilitating Generalized Advantage Estimation (GAE) to stabilize learning by smoothing reward signals and reducing variance. The maximum number of steps per episode is determined by the maximum number of VNFs but may be adjusted to a smaller value based on the network environment

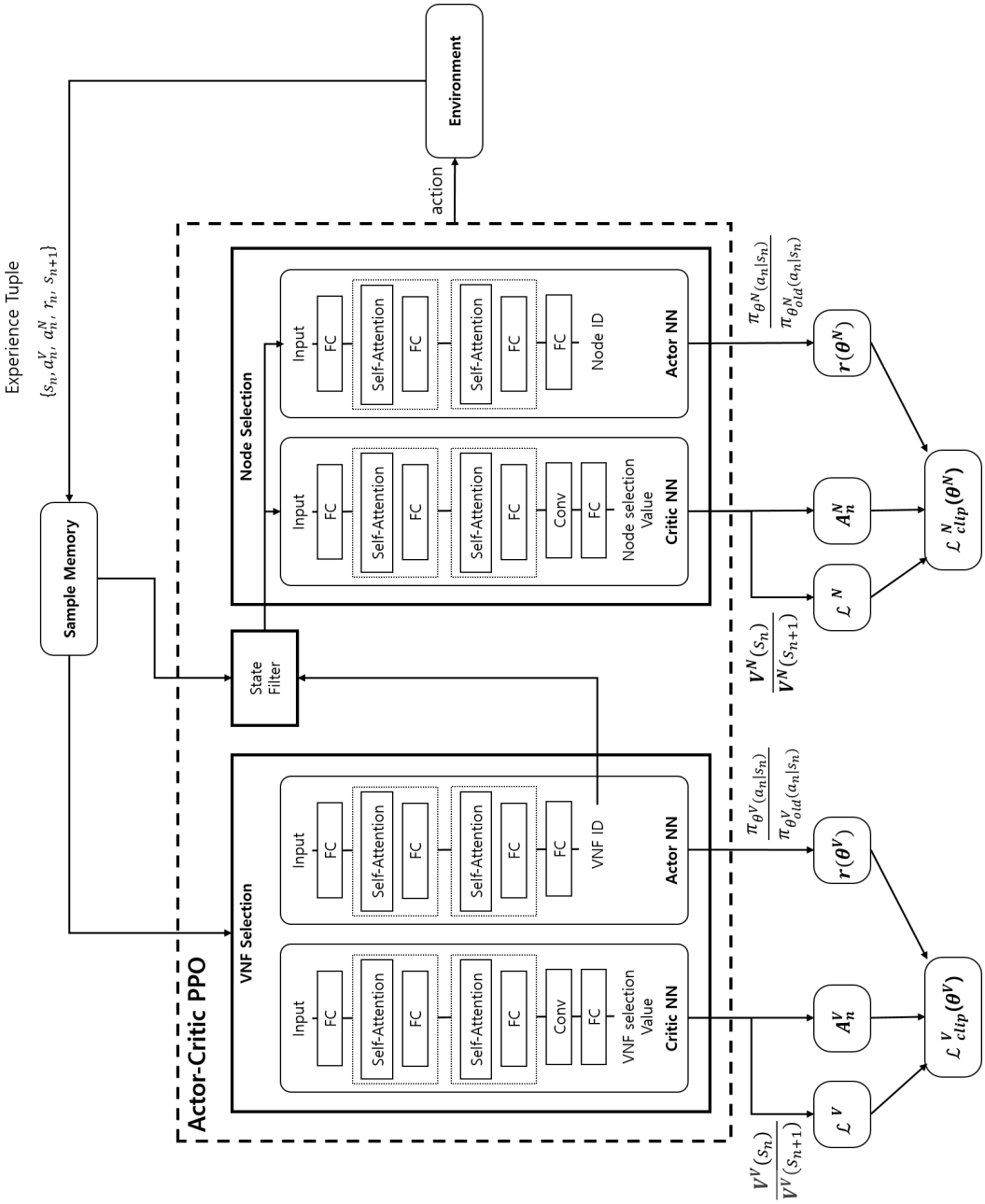

Figure 4.6: RL model of the VNF consolidation

	VNF selection	Node selection
Learning rate of actor network	0.001	0.001
Learning rate of critic network	0.001	0.001
Optimizer of actor network	Adam	Adam
Optimizer of critic network	Adam	Adam
Policy clip range	0.1	0.1
Entropy loss weight	0.01	0.01
Discount factor	0.99	
GAE parameter	0.97	
Number of wokers	8	
Size of batch	32	
Update epochs	80	
Maximum number of episode	50,000	
Maximum size of memory	200	

Table 4.5: Hyperparemter of VNF Consolidation

V. Evaluation

To evaluate the proposed NFV-MANO system, traffic was generated using network information from the Internet2 [1]. Over one week, the total network bandwidth in the Internet2 topology was aggregated and averaged to measure the variations in bandwidth. Figure 5.1 illustrates the bandwidth fluctuations in the Internet2 topology. Following these fluctuations, new network services were created, or additional flows were generated to simulate real network bandwidth changes. During traffic generation, if the corresponding source and destination clients do not exist in the network, the respective client VNFs are deployed using NFVI and NFVI. The increase or decrease in traffic bandwidth is managed at the flow or service levels.

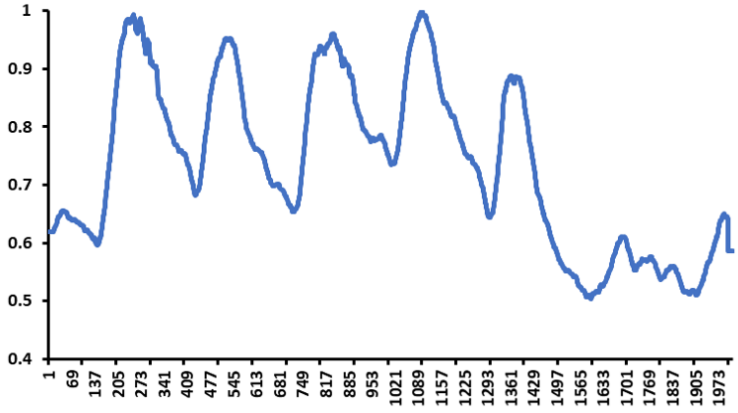

Figure 5.1: The traffic pattern of Internet2

[1] http://abilene.internet2.edu/

5.1 VNF Deployment

The deployment function installs VNFs to support new network services in the precise definition. Thus, it does not account for the traffic flow of existing services but instead concentrates solely on scenarios involving the addition of new services. Figure 5.2 illustrates the performance of the deployment model under the learner controller. The learner controller specifies the variations in the learning network environment and the number of prospective services to be considered. With an increase in the number of considered services, the model's performance improves, achieving 100% from 97.5% as the number of services increases from 10 to 10,000. Training on the entire traffic pattern with 10,000 SFCRs, the developed model seems capable of replacing ILP. However, despite achieving high performance on the test set evaluation, the model reaches a maximum of 92.3% when learning with newly generated traffic in diverse environments. The reasons are as follows. When considering traffic patterns, network services are requested randomly. In this case, the initially installed network service preoccupies network resources, affecting the installation location of subsequent services. This process is repeated, and since the learning model only learns the ILP results, any slight change in the network environment negatively impacts the learned model. In other words, even if a high-performance model is created, if the traffic data and deployment installation pattern slightly change, the model becomes inapplicable. To mitigate this issue, we considered 100,000 requested SFCRs, however, it is still practically impossible to cover all potential scenarios, including those with already installed SFCRs. Fortunately, in this thesis, the model is pre-trained for future environments, and a suitable learning model is applied using cosine similarity, mitigating the aforementioned issue.

Figure 5.3 shows the performance of the learning model according to cosine similarity. The graph shows noticeable performance degradation at lower similarity levels, with minimal degradation at higher similarity levels. The model ensures consistent performance when the cosine similarity between the pre-trained network environment

and the current environment remains within a specified range. However, performance significantly declines when this similarity drops below a specific threshold. This is especially noticeable during significant resource changes in specific nodes. For instance, when all nodes have ample resources for multiple VNF installations, the model performs well. However, in resource-constrained environments, the model may make incorrect decisions. Training across a broad range of environments prevented significant performance drops even with low cosine similarity. However, it's practically impossible to train for every conceivable environment, and the model continues to face challenges in adapting to topological changes.

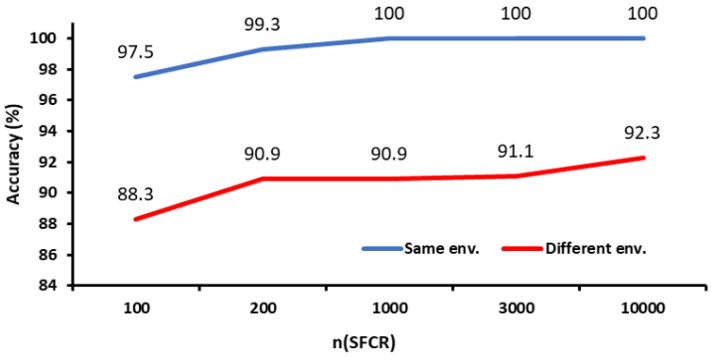

Figure 5.2: VNF deployment accuracy based on the number of SFCR

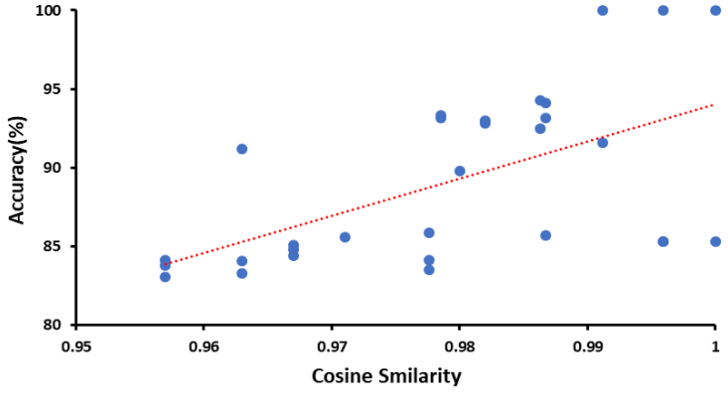

Figure 5.3: Performance difference of VNF deployment based on cosine similarity

5.2 Auto-scaling

The auto-scaling function installs VNFs and chaining SFCs. Each SFC requires individual learning, and the learner controller supports scaling learning for each type, enabling the creation of services by combining specific numbers of source and destination clients. Figure 5.4 illustrates the reward of the auto-scaling module per epoch.The results suggest that SFCs with fewer tiers converge more quickly in determining scaling actions.

The delay with and without scaling applied to various SFCs was measured. The SFC types were consistently composed of firewall, flow monitor, DPI, IDS, and proxy, with different VNF instance locations and varying source and destination nodes to create differences in routing paths. To account for measurement errors, 20 measurements were conducted, and the results are presented in Figure 5.5.

The figure shows that the overall delay decreases when scaling is applied. Also, it shows that the scaling is more significant when routing paths are simpler. This indicates that scaling improves service performance by resolving resource shortages, but additional improvements are needed to address the delays caused by unnecessary routing paths.

Utilizing the learner controller, additional SFCs and network environments are generated and trained. Figure 5.6 compares the learning performance based on the cosine similarity of SFCs. It shows that models with low similarity exhibit lower performance, while those with high similarity achieve better learning outcomes. Figure 5.7 illustrates a performance comparison based on the cosine similarity of the network environment. It shows that learning progresses relatively smoothly even at lower similarity levels. This is because the primary learning target of RL is SFC information rather than network information, and most interactions with the environment are executed as macro-actions.

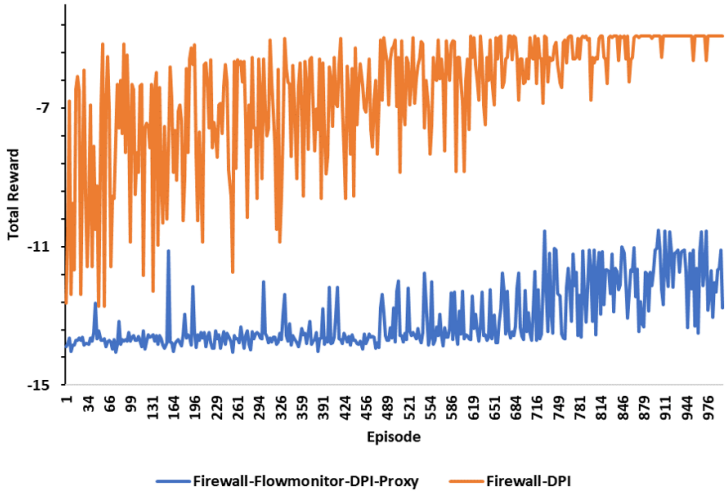

Figure 5.4: Reward obtained per each episode

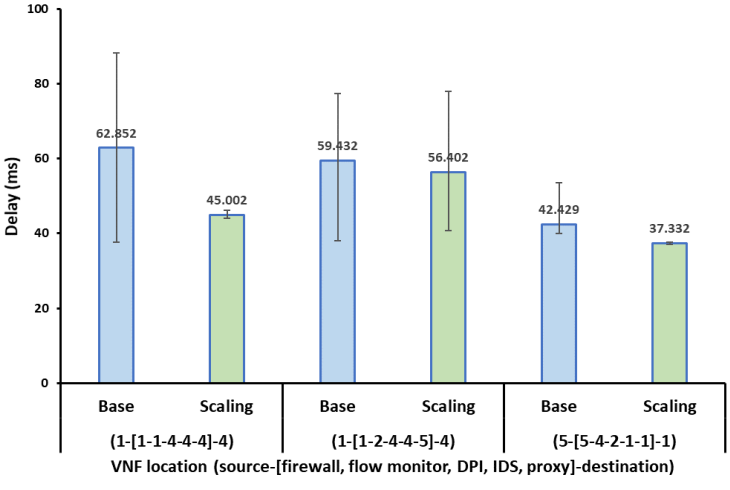

Figure 5.5: Delay difference of auto-scaling based on SFCs

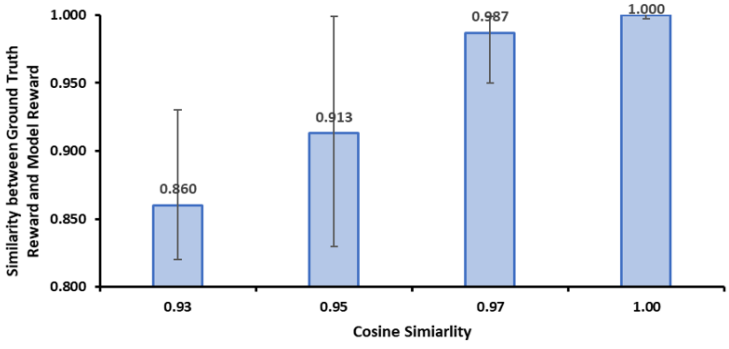

Figure 5.6: Performance difference of auto-scaling based on cosine similarity for SFC

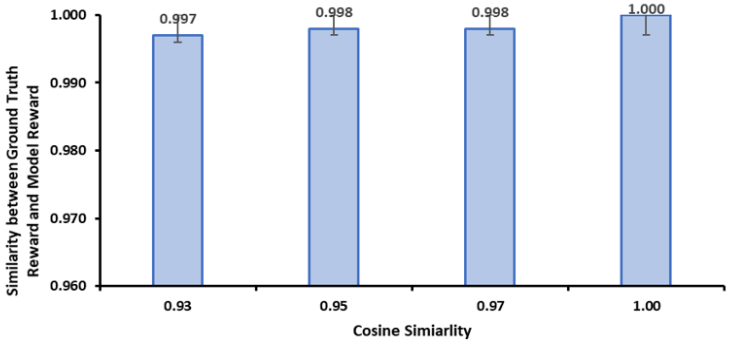

Figure 5.7: Performance difference of auto-scaling based on cosine similarity for network environment

5.3 Service Function Chaining

The service function chaining function underwent similar experimentation to auto-scaling. Utilizing the learner controller supports learning to scale each type and facilitates service creation by combining specific numbers of source and destination clients. Unlike auto-scaling, which responds to traffic load for scaling, service function chaining primarily deals with issues caused by specific VNFs, such as scenarios involving injected stress into installed VNFs.

The delay with and without chaining was measured. The SFC types were configured identically to those with auto-scaling, and the VNF instances were arranged to create differences in routing paths. Using stress-ng, arbitrary VNFs were overloaded, and additional VNFs were randomly placed on nodes to enable chaining. To account for measurement errors, 20 measurements were conducted. The results are presented in Figure 5.8. The figure shows that the overall delay decreases when chaining is applied. Particularly, chaining significantly reduced the delay for some SFCs by alleviating resource overload and ensuring the paths were the shortest possible. However, if there is no available VNF for chaining or no guarantee of the shortest path, the delay is only partially reduced.

By utilizing the controller, additional SFCs and network environments were generated for learning. Figure 5.9 compares the learning performance based on the cosine similarity of SFCs. It demonstrates that models with lower similarity exhibit poor performance, similar to the auto-scaling model. Figure 5.10 illustrates a performance comparison based on the cosine similarity of the network environment. It shows that learning generally proceeds smoothly even at lower similarity levels, except during extreme network state changes where performance degradation occurs.

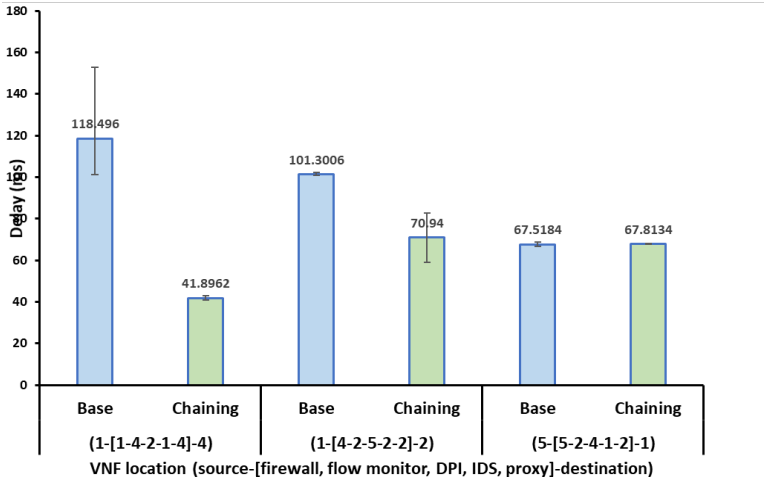

Figure 5.8: Delay difference of service function chaining based on SFCs

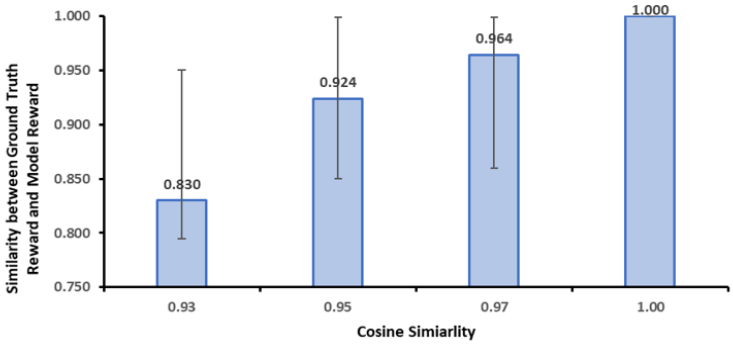

Figure 5.9: Performance difference of service function chaining based on cosine similarity for SFC

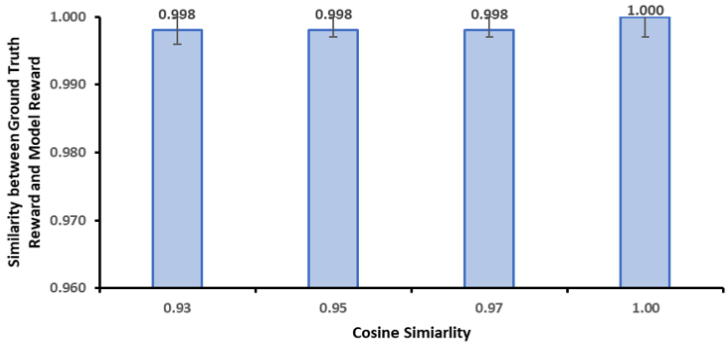

Figure 5.10: Performance difference of service function chaining based on cosine similarity for network environment

5.4 Consolidation

The consolidation learning model trained a separate model for each edge network. The learner controller created a learning environment that can simulate multiple VNFs and SFCs for edge networks. Figure 5.11 illustrates the reward per epoch during consolidation learning with 12 VNFs. Figure 5.12 depicts the performance of the trained model compared to the ground truth for varying numbers of VNFs. According to the graph, the consolidation function does not achieve the correct solution for certain learning environments. For instance, the performance in environments with an odd number of VNFs does not generally show favorable results. This outcome arises because the maximum step size has not been appropriately defined. The current step size is defined as the number of VNFs multiplied by two, which enforces an even number of migrations. Consequently, this results in the execution of unnecessary actions, potentially leading to oscillations or other issues.

Figure 5.13 illustrates the model performance based on cosine similarity. The graph indicates that the consolidation model does not perform as smoothly as other management models in scenarios where highly similar models are absent. This disparity arises because, unlike other models that use macro actions, the consolidation

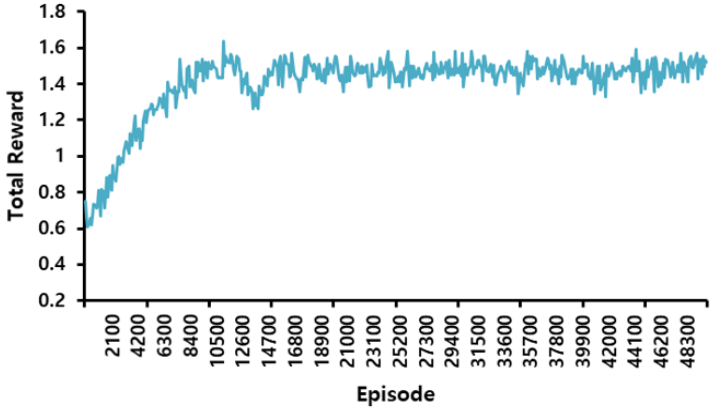

Figure 5.11: Reward obtained per each episode

model learns all individual actions, thereby increasing its reliance on surrounding environmental data. Moreover, the PPO structure requires hyperparameter adjustments with changes in learning data, and failure to make these adjustments can lead to performance issues.

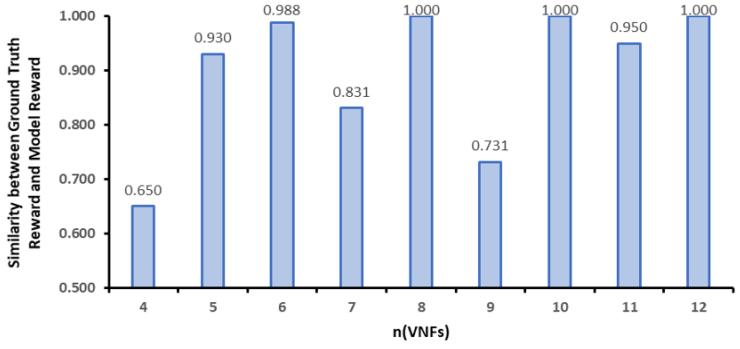

Figure 5.12: Performance difference of consolidation based on the number of VNFs

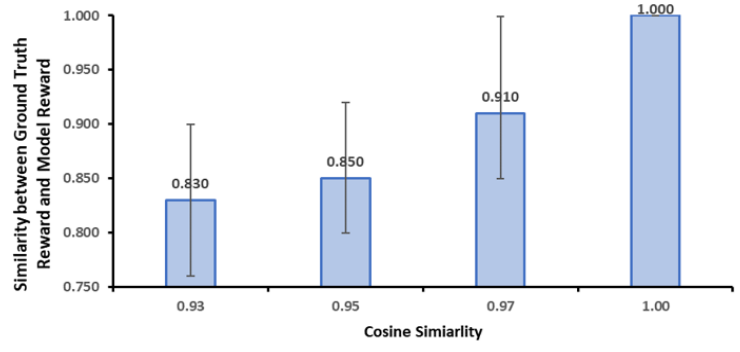

Figure 5.13: Performance difference of consolidation based on cosine similarity

VI. Conclusion

6.1 Summary

This thesis introduced and developed an AI-based NFV MANO. The system aims to enhance the efficiency and automation of managing VNFs in an NFV environment. This includes tasks such as VNF deployment, auto-scaling, service function chaining, and consolidation. The major contributions of this research are detailed below.

Firstly, the AI-based NFV management system was designed to automate the data collection, preprocessing, and the evaluation of machine learning models. This automation process was applied to various network environments to ensure that the system could adapt and perform optimally across different scenarios. The study demonstrated high performance by employing and training diverse machine learning models specifically tailored for four key VNF management policies: VNF deployment, service function chaining, auto-scaling, and consolidation.

Secondly, the research introduced compartmentalized management policies to minimize conflicts among different NFV management functions. By leveraging network domain knowledge, the complexity of the management problems was reduced, making it easier for machine learning models to learn and implement effective management strategies. This compartmentalization ensures that each management function operates within a precise scope, thereby reducing the likelihood of policy conflicts.

Thirdly, a proactive learning system was developed. This system enables the NFV MANO to conduct learning in advance for various network environments. By pre-training machine learning models and loading them in real-time network environments, the system ensures that optimal management policies are always in place. This proactive approach minimizes the time required for model training during actual network operation, thereby enhancing the system's responsiveness and efficiency.

Fourthly, the system orchestrates management policies to prevent conflicts and competition among different management functions. By orchestrating these policies effectively, the system prevents any degradation in overall system performance, ensuring that all management functions work harmoniously to achieve the desired network outcomes.

Lastly, the thesis developed a digital twin system to replicate an NFV environment. This digital twin creates a simulation environment where machine learning models can be trained safely without risking actual network failures due to incorrect policies. By learning in this simulated environment, the system can proactively prepare for various network conditions, ensuring robust and reliable performance in real-world scenarios.

6.2 Future Work

While the proposed AI-based NFV MANO system shows significant promise, further research is necessary to address the complexities and dynamic nature of real-world network environments. Several future research directions are identified.

One key area for future work is the development of scalable learning models. The current system is trained for specific network environments, but there is a need for models that can adapt to a broader range of network conditions. Developing scalable models will enhance the system's flexibility and applicability to diverse network scenarios. Another important direction is the implementation of real-time learning and adaptation mechanisms. As network environments can change rapidly, it is crucial for the learning models to continuously learn and adapt in real-time. This capability will ensure that the system remains effective even as network conditions evolve.

Enhancing the security and reliability of the AI-based network management system is also a critical area for future research. Analyzing the potential security vulnerabilities and developing robust security mechanisms to mitigate these risks will be essential for ensuring the safe and reliable operation of the system. Additionally,

it will be important to test the proposed system in a variety of real-world network scenarios. This testing will validate the system's performance and effectiveness, providing insights into how the system can be further improved and refined for practical deployment.

Finally, further research should focus on the interactions between different NFV management policies. By conducting a detailed analysis of these interactions, optimal orchestration strategies can be developed to minimize conflicts and enhance overall system performance.

This thesis has highlighted the potential of AI-based NFV Management and Orchestration systems. Continued research in these areas is expected to lead to the development of even more advanced and effective network management solutions, further advancing the field of network virtualization and automation.

요 약 문

이 논문은 NFV 관리 및 오케스트레이션 시스템(NFV MANO)을 AI 기반으로 도입하고 개발하였다. 이 시스템은 NFV 환경에서 가상 네트워크 기능(VNF)의 관리를 효율적으로 자동화하는 것을 목표로 한다. 여기에는 VNF 배포, 자동 스케일링, 서비스 기능 체이닝, VNF 통합 등의 작업이 포함된다. 본 연구의 주요 기여는 아래와 같다.

첫째, AI 기반 NFV 관리 시스템은 데이터 수집, 전처리 및 머신러닝 모델의 평가를 자동화하도록 설계되었다 이 자동화 프로세스는 다양한 네트워크 환경에 적용되어 시스템이 여러 시나리오에서 적응하고 최적의 성능을 발휘할 수 있도록 보장한다. 제안한 시스템은 네 가지 주요 VNF 관리 정책(VNF 배포, 서비스 기능 체이닝, 자동 스케일링, VNF 통합)에 특화된 다양한 머신러닝 모델을 적용하고 훈련하여 높은 성능을 입증하였다.

둘째, 다양한 NFV 관리 기능 간의 충돌을 최소화하기 위해 관리 정책을 구획화하였다. 또한, 네트워크 도메인 지식을 활용하여 관리 문제의 복잡성을 줄였고 이를 통해 머신러닝 모델이 효과적인 관리 전략을 학습할 수 있었다. 구획화 과정에서는 각 관리 기능이 좁게 정의된 범위 내에서 운영되도록 하여 정책 충돌의 가능성을 줄였다.

셋째, 사전 학습 시스템을 개발하였다. 제안하는 시스템은 NFV MANO가 다양한 네트워크 환경에 대해 사전 학습을 수행할 수 있게 한다. 머신러닝 모델을 다양한 네트워크에 대해 사전 훈련을 진행하고 이를 실시간 네트워크 환경에 로드함으로

써 시스템에 항상 최적의 관리 정책이 적용되도록 보장하였다. 이와 같은 사전 대응 접근 방식은 실제 네트워크 운영 중에 모델 훈련에 필요한 시간을 최소화하여 시스템의 반응성과 효율성을 향상시킨다.

넷째, 연구는 다양한 관리 기능 간의 충돌과 경쟁을 피하기 위해 관리 정책의 오케스트레이션을 진행하였다. 정책을 효과적으로 오케스트레이션함으로써 전체 시스템의 성능 저하를 방지하고 모든 관리 기능이 원하는 네트워크 결과를 달성하기 위해 조화롭게 작동하도록 하였다.

마지막으로 NFV 환경을 복제하는 디지털 트윈 시스템을 개발하였다. 개발한 디지털 트윈은 머신러닝 모델을 안전하게 훈련할 수 있는 시뮬레이션 환경을 조성하여 잘못된 정책으로 인한 실제 네트워크의 실패를 방지한다. 또한, 시뮬레이션 환경에서 학습함으로써 시스템은 다양한 네트워크 조건에 대해 사전 대비를 할 수 있어 실제 환경에서 견고하고 신뢰할 수 있는 성능을 보장하였다.

References

[1] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2014.

[2] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications surveys & tutorials*, 18(1):236–262, 2015.

[3] Joel Halpern and Carlos Pignataro. Service function chaining (sfc) architecture. Technical report, 2015.

[4] Chafika Benzaid and Tarik Taleb. Ai-driven zero touch network and service management in 5g and beyond: Challenges and research directions. *Ieee Network*, 34(2):186–194, 2020.

[5] Maulshree Singh, Evert Fuenmayor, Eoin P Hinchy, Yuansong Qiao, Niall Murray, and Declan Devine. Digital twin: Origin to future. *Applied System Innovation*, 4(2):36, 2021.

[6] GSNFV ETSI. 001, network functions virtualisation (nfv); use cases. *Group Specification*, 2017.

[7] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

[8] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017.

[9] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[10] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[11] Jianqing Fan, Zhaoran Wang, Yuchen Xie, and Zhuoran Yang. A theoretical analysis of deep q-learning. In *Learning for dynamics and control*, pages 486–489. PMLR, 2020.

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[13] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[14] Tim Van Erven and Peter Harremos. Rényi divergence and kullback-leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014.

[15] Stanislav Lange, Hee-Gon Kim, Se-Yeon Jeong, Heeyoul Choi, Jae-Hyung Yoo, and James Won-Ki Hong. Machine learning-based prediction of vnf deployment decisions in dynamic networks. In *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 1–6. IEEE, 2019.

[16] Erin LeDell and Sebastien Poirier. H2o automl: Scalable automatic machine learning. In *Proceedings of the AutoML Workshop at ICML*, volume 2020. ICML San Diego, CA, USA, 2020.

[17] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[18] Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 2013.

[19] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[20] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63:3–42, 2006.

[21] Doyoung Lee, Jae-Hyoung Yoo, and James Won-Ki Hong. Deep q-networks based auto-scaling for service function chaining. In *2020 16th International Conference on Network and Service Management (CNSM)*, pages 1–9. IEEE, 2020.

[22] DongNyeong Heo, Stanislav Lange, Hee-Gon Kim, and Heeyoul Choi. Graph neural network based service function chaining for automatic network control. In *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 7–12. IEEE, 2020.

[23] Jing Chen, Jia Chen, and Hongke Zhang. Drl-qor: Deep reinforcement learning-based qos/qoe-aware adaptive online orchestration in nfv-enabled networks. *IEEE Transactions on Network and Service Management*, 18(2):1758–1774, 2021.

[24] Li Xia and Qing-Shan Jia. Parameterized markov decision process and its application to service rate control. *Automatica*, 54:29–35, 2015.

[25] Haojun Huang, Cheng Zeng, Yangmin Zhao, Geyong Min, Yingying Zhu, Wang Miao, and Jia Hu. Scalable orchestration of service function chains in nfv-enabled networks: A federated reinforcement learning approach. *IEEE Journal on Selected Areas in Communications*, 39(8):2558–2571, 2021.

[26] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE conference on computer communications*, pages 1698–1707. IEEE, 2020.

[27] Eui-Dong Jeong, Jae-Hyoung Yoo, and James Won-Ki Hong. Sdn lullaby: Vm consolidation for sdn using transformer-based deep reinforcement learning. In *2023 19th International Conference on Network and Service Management (CNSM)*, pages 1–5. IEEE, 2023.

[28] Joao Soares, Miguel Dias, Jorge Carapinha, Bruno Parreira, and Susana Sargento. Cloud4nfv: A platform for virtual network functions. In *2014 IEEE 3Rd international conference on cloud networking (cloudnet)*, pages 288–293. IEEE, 2014.

[29] Wenyu Shen, Masahiro Yoshida, Kenji Minato, and Wataru Imajuku. vconductor: An enabler for achieving virtual network integration as a service. *IEEE Communications Magazine*, 53(2):116–124, 2015.

[30] Omar Sefraoui, Mohammed Aissaoui, Mohsine Eleuldj, et al. Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, 55(3):38–42, 2012.

[31] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. kvm: the linux virtual machine monitor. In *Proceedings of the Linux symposium*, volume 1, pages 225–230. Dttawa, Dntorio, Canada, 2007.

[32] Jan Medved, Robert Varga, Anton Tkacik, and Ken Gray. Opendaylight: Towards a model-driven sdn controller architecture. In *Proceeding of IEEE international*

symposium on a world of wireless, mobile and multimedia networks 2014, pages 1–6. IEEE, 2014.

[33] Marouen Mechtri, Chaima Ghribi, Oussama Soualah, and Djamal Zeghlache. Etso: End-to-end sfc orchestration framework. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 903–904. IEEE, 2017.

[34] OASIS Standard. Topology and orchestration specification for cloud applications version 1.0, 2013.

[35] Marouen Mechtri, Chaima Ghribi, and Djamal Zeghlache. Vnf placement and chaining in distributed cloud. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 376–383. IEEE, 2016.

[36] Md Faizul Bari, Shihabur Rahman Chowdhury, Reaz Ahmed, and Raouf Boutaba. On orchestrating virtual network functions. In *2015 11th international conference on network and service management (CNSM)*, pages 50–56. IEEE, 2015.

[37] Chaima Ghribi, Marouen Mechtri, and Djamal Zeghlache. A dynamic programming algorithm for joint vnf placement and chaining. In *Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking*, pages 19–24, 2016.

[38] Oussama Soualah, Marouen Mechtri, Chaima Ghribi, and Djamal Zeghlache. An efficient algorithm for virtual network function placement and chaining. In *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 647–652. IEEE, 2017.

[39] Oussama Soualah, Marouen Mechtri, Chaima Ghribi, and Djamal Zeghlache. A link failure recovery algorithm for virtual network function chaining. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 213–221. IEEE, 2017.

[40] Yue Wang, Ray Forbes, Uri Elzur, John Strassner, Antonio Gamelas, Haining Wang, Shucheng Liu, Luca Pesando, Xiangfeng Yuan, and Shengming Cai. From design to practice: Etsi eni reference architecture and instantiation for network management and orchestration using artificial intelligence. *IEEE Communications Standards Magazine*, 4(3):38–45, 2020.

[41] David M Gutierrez-Estevez, Marco Gramaglia, Antonio De Domenico, Ghina Dandachi, Sina Khatibi, Dimitris Tsolkas, Irina Balan, Andres Garcia-Saavedra, Uri Elzur, and Yue Wang. Artificial intelligence for elastic management and orchestration of 5g networks. *IEEE wireless communications*, 26(5):134–141, 2019.

[42] Hee-Gon Kim, Suhyun Park, Stanislav Lange, Doyoung Lee, Dongnyeong Heo, Heeyoul Choi, Jae-Hyoung Yoo, and James Won-Ki Hong. Graph neural network-based virtual network function management. In *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 13–18. IEEE, 2020.

[43] Faizul Bari, Shihabur Rahman Chowdhury, Reaz Ahmed, Raouf Boutaba, and Otto Carlos Muniz Bandeira Duarte. Orchestrating virtualized network functions. *IEEE Transactions on Network and Service Management*, 13(4):725–739, 2016.

[44] Christian Bliek1ú, Pierre Bonami, and Andrea Lodi. Solving mixed-integer quadratic programming problems with ibm-cplex: a progress report. In *Proceedings of the twenty-sixth RAMP symposium*, pages 16–17, 2014.

Acknowledgements

학위 과정을 마무리하면서 지난 시간들을 돌아보니 감회가 정말 새롭습니다. 부족한 저를 좋은 방향으로 지도하여 주신 홍원기 교수님, 유재형 교수님께 깊이 감사를 드립니다. 좋은 연구 환경 아래에서 교수님들의 훌륭한 지도로 열심히 배우고 성장할 수 있는 소중한 시간을 보낼수 있었습니다. 특히나 교수님 덕분에 다양한 연구원 분들과도 협력 연구를 진행하고 많이 발전할 수 있는 기회를 얻었습니다. 연구 외적으로도 여러 경험을 할 수 있는 기회를 제공하여주시고 연구자뿐만 아니라 사회인으로서의 필요한 덕목도 배울수 있었습니다.

처음 대학교에 입학하여 포항에 온 뒤로 박사 학위를 마치기까지 오랜 시간을 보냈습니다. 그 기간 동안 저에게 늘 기운을 주시고 사랑한다고 말씀해주신 아버지, 어머니께 진심으로 감사드립니다. 아버지, 어머니 사랑합니다. 항상 믿어주시고 격려해주시는 다른 친지분들께도 고마움을 전하고 싶습니다. 오랜 은사이신 손정순 선생님과 이언정 선생님께 큰 감사인사를 드리며 대학교부터 시작하여 11년의 시간을 같이 보내준 경용이와 휘수, 원준이에게도 고마움을 전하고 싶습니다.

DPNM 연구실은 저에게 좋은 기억만을 남겨주었습니다. 신입생부터 최고참이 될 때까지, 모든 선후배 분들과 재밌고 즐거웠던 기억이 가득합니다. 연구실 선배였던 세연이형과 도영이형에게 많은 도움을 받아 고마움을 전하며 특히, 연구실 선배이자 룸메이트로 긴 시간을 함께 보낸 경찬이형에게 다시 한번 고마움을 전합니다. 연구를 같이 진행한 석현이와 의동이에게도 고마움을 표하며, 이외에도 학위과정 동안 함께 연구했던 모든 선후배님들께 감사를 드립니다. 앞으로도 좋은 인연이 계속해서 이어지기를 바랍니다.

제 인생에서 지난 6년의 시간은 잊지 못할 기억으로 남을 것 같습니다. 내외의 뛰어난 연구원 분들과 협업을 통해 얻은 성취감과 더불어 좋은 사람들과의 즐거웠던 여러 시간들이 늘 떠오를 것 같습니다. 앞으로 연구실에서의 경험과 교수님들의 가르침을 발판삼아 더욱 발전하고 제가 받은 은혜에 보답하여 다른 사람에게 도움이 될 수 있는 사람이 되겠습니다. 감사합니다.

Curriculum Vitae

Personal Information

Name : **Hee Gon Kim**

Position : Ph.D. student

Laboratory : Distributed Processing & Network Management (DPNM) Lab.

Department : Computer Science and Engineering

Education

2018. 03. – 2024. 08. Department of Computer Science and Engineering, Pohang University of Science and Technology (Ph.D.)

2013. 03. – 2017. 08. Department of Computer Science and Engineering, Pohang University of Science and Technology (B.S.)

Research Areas of Interest

Software-Defined Networking (SDN), Network Function Virtualization (NFV), Cloud Computing, Artificial Intelligence-based Network Management

Research / Project Experiences

1. **Development of core technologies for programmable switches in multi-service networks**

   Funded by the Institute for Information & Communications Technology Planning & evaluation (IITP) (2018 – 2020)

   This project aims to support multi-service networks based on programmable switches. The research goals focus on extending P4 languages and defining a new switch machine model, compiler, and multi-service network architecture. My contribution to this project is surveying knowledge-defined networking and P4. Additionally, intent-based networking has been studied since the beginning of this project.

2. **Development of blockchain transaction monitoring and analysis technology**

   Funded by Institute for Information & Communications Technology Planning & evaluation (2018 – 2020)

   This research project aims to develop a blockchain transaction monitoring and analysis system for collecting and analyzing transactions from various blockchains (such as Bitcoin, Ethereum). The system will be used for detecting illegal transactions, identifying security attacks, conducting blockchain forensics, and managing private blockchain operations. Basic analyses include blockchain network performance analysis, blockchain node performance analysis, DDoS detection analysis, and smart contract security vulnerability analysis. My contribution to this project involves developing and analyzing a blockchain explorer. This includes collecting data on both legal and illegal transactions and using machine learning to identify and distinguish the characteristics of these transactions.

3. **Development of internet infrastructure system technologies and R&D human resources**

   Funded by Institute for Information & Communications Technology Planning & evaluation (2018 – 2022)

   This research project aims to develop open-source SDN/NFV-based networking software. It also focuses on validating the development results and establishing test environments using internet infrastructure systems like KREONET/KOREN. My contribution to this project involves developing an OpenStack-based NFV system, including AI-based VNF deployment.

4. **Development of virtual network management technology based on artificial intelligence**

   Funded by Institute for Information & Communications Technology Planning & evaluation (IITP) (2018 – 2023)

   This research project aims to study virtual network management, including the development of AI-based algorithms essential for VNF life-cycle management to monitor and control resources in the NFV environment in real-time. My contribution to this project involved building an OpenStack-based NFV testbed and implementing a PoC system for machine learning-based NFV management. I designed and developed algorithms and modules for VNF deployment, auto-scaling, service function chaining, VNF migration, and VNF consolidation. Additionally, I designed and developed the overall orchestration model, ensuring that each management function model could automatically optimize the virtual network environment.

5. **PSAI Industry-academic joint research and education program**

   Funded by Institute for Information & Communications Technology Planning & Evaluation (2020 – 2021)

   This research project aims to provide advanced AI education for outstanding master and doctoral students at Stanford University and to conduct collabora-

tive research focused on practical projects. Due to the COVID-19 pandemic, the classes and research were conducted remotely. My contribution to this project involved studying reinforcement learning-based routing algorithms and hierarchical graph neural networks for network data abstraction. As part of this project, I completed the AI course at Stanford and conducted collaborative research with the University of Toronto.

6. **Development of traffic engineering based on artificial intelligence**
   Funded by Samsung (2020 – 2022)
   This research project aims to develop an artificial intelligence (AI)-based fast routing algorithm using network status information monitored in the segment routing environment. This algorithm not only satisfies service requirements (e.g., throughput, latency, and packet loss) but also handles network failures. My contribution to this research was to develop an algorithm for selecting middle points in a segment routing environment using deep reinforcement learning models. Paths were generated for incoming traffic in a Wide Area Network (WAN) environment in real-time, ensuring appropriate distribution of network load. On/off-line learning experiments were conducted using a model developed with deep double dueling Q-network and actor-critic network, which enhanced the minimum available bandwidth of network links.

7. **Development of AI network traffic controlling system based on SDN for ultra-low latency network service**
   Funded by Ministry of Trade, Industry and Energy (2020 – 2023)
   This research project aims to develop an SDN-based ultra-low latency network service framework and IoT applications. It also focuses on supporting stabilization and integration through an SNMP-based management system. Additionally, it aims to study machine learning algorithms and data models for network management to detect network failures and intrusions. My contribution to this project involves developing and modeling network failure and intrusion de-

tection algorithms. This includes developing models using public datasets and conducting tests and validations in real network scenarios.

8. **Development of network anomaly detection technology based on natural language processing and machine learning**
   Funded by Samsung (2022 – 2024)
   The research project aims to develop technology for real-time detection of abnormal operational states in routers/switches that comprise a network, utilizing natural language processing and machine learning algorithms to analyze logs, NETCONF, SNMP messages, etc. In this study, a log parser was developed to extract log patterns from input logs and transform them into log events through pattern analysis. The parser extracts identical patterns in logs based on a pre-established dictionary and converts log data into a flow of event numbers. Using this approach, an LSTM (Long Short-Term Memory) based model was trained to predict the next event number by learning normal event flows. A method was proposed to quantify the risk level of each log by using TF-IDF to numerically assess the importance of each word and its frequency in normal occurrences. My contribution to this project involved designing a log risk quantification model based on TF-IDF.

9. **Development of integrated intelligent plane technology for 6G networks**
   Funded by Institute for Information & Communications Technology Planning & evaluation (2024 – 2024)
   This research project aims to develop intelligent plane technology that provides AI-based optimal inference and control policies through efficient end-to-end data collection in a 6G integrated domain (RAN/Transport/Core/Data/OAM) mobile network. It focuses on acquiring and processing 6G integrated domain data, developing inference coordination technology between domains, constructing learning environments, and optimizing control policies for automation. My contribution to this project involves designing the Management Data Analytics

Function (MDAF).

International Journal Papers

1. Stanislav Lange, Nguyen Van Tu, Se-Yeon Jung, Do-Young Lee, **Hee-Gon Kim**, Jibum Hong, Jae-Hyoung Yoo, James Won-Ki Hong, ”A Network Intelligence Architecture for Efficient VNF Lifecycle Management”, IEEE Transactions on Network and Service Management (TNSM) (SCIE), August 2020.
2. **Hee-Gon Kim**, Suhyun Park, Stanislav Lange, Doyoung Lee, Dongnyeong Heo, Heeyoul Choi, Jae-Hyoung Yoo, James Won-Ki Hong, ”Graph neural network-based virtual network function deployment optimization”, International Journal of Network Management (IJNM) (SCIE), May 2021.

Domestic Journal Papers

1. **Hee-Gon Kim**, Do-Young Lee, Jae-Hyung Yoo, James Won-Ki Hong, A Machine Learning-based Method for Virtual Network Function Resource Demand Prediction, KNOM Review, Vol. 21, No. 2, December 2018. pp. 1-9.
2. Suhyun Park, **Hee-Gon Kim**, Jibum Hong, Jae-Hyung Yoo, James Won-Ki Hong, Machine Learning-based Optimal VNF Deployment Prediction, KNOM Review Vol. 23, No. 1, August 2020, pp. 34-42.

International Conference Papers

1. Chaehyeon Lee, **Heegon Kim**, Sajan Maharjan, Kyungchan Ko and James Won-Ki Hong, "Blockchain Explorer based on RPC-based Monitoring System", IEEE International Conference on Blockchain and Cryptocurrency (ICBC 2019), Seoul, Korea, May. 16, 2019.

2. **Heegon Kim**, Doyoung Lee, Seyeon Jeong, Heeyoul Choi, Jae-Hyoung Yoo, James W. Hong, "A Machine Learning-based Method for Virtual Network Function Resource Demand Prediction", 5th IEEE Conference on Network Softwarization (Netsoft 2019), Paris, France, June 24-28, 2019.

3. **Heegon Kim**, Seyeon Jeong, Doyoung Lee, Heeyoul Choi, Jae-Hyoung Yoo, James W. Hong, "A Deep Learning Approach to VNF Resource Prediction using Correlation between VNFs", 2nd International Workshop on Emerging Trends in Softwarized Networks (ETSN 2019), Paris, France, June 28, 2019.

4. Stanislav Lange, **Heegon Kim**, Seyeon Jeong, Heeyoul Choi, Jae-Hyoung Yoo, James W. Hong, "Machine Learning-based Prediction of VNF Deployment Decisions in Dynamic Networks", The 20th Asia-Pacific Network Operations and Management Symposium (APNOMS 2019), Matsue, Japan, Sep. 18-20, 2019.

5. Seyeon Jeong, **Heegon Kim**, Jae-Hyoung Yoo, James W. Hong, "Machine Learning Based Link State Aware Service Function Chaining", The 20th Asia-Pacific Network Operations and Management Symposium (APNOMS 2019), Matsue, Japan, Sep. 18-20, 2019.

6. Stanislav Lange, **Heegon Kim**, Seyeon Jeong, Heeyoul Choi, Jae-Hyoung Yoo, James W. Hong, "Predicting VNF Deployment Decisions under Dynamically Changing Network Conditions", 15th International Conference on Network and Service Management (CNSM 2019), Halifax, Canada, Oct. 21-25, 2019.

7. **Hee-Gon Kim**, Suhyun Park, Stanislav Langey, Doyoung Lee, Dongnyeong Heo, Heeyoul Choi, Jae-Hyoung Yoo, James Won-Ki Hong, "Graph Neural

Network-based Virtual Network Function Management", The 21st Asia-Pacific Network Operations and Management Symposium (APNOMS 2020), Daegu, Korea, Sep. 23-25, 2020 **(Student Paper Award)**.

8. Suhyun Park, **Hee-Gon Kim**, Jibum Hong, Stanislav Langey, Jae-Hyoung Yoo, James Won-Ki Hong, "Machine Learning-based Optimal VNF Deployment", The 21st Asia-Pacific Network Operations and Management Symposium (APNOMS 2020), Daegu, Korea, Sep. 23-25, 2020.

9. DongNyeong Heo, Stanislav Lange, **Hee-Gon Kim**, Heeyoul Choi, "Graph Neural Network based Service Function Chaining for Automatic Network Control", The 21st Asia-Pacific Network Operations and Management Symposium (APNOMS 2020), Daegu, Korea, Sep. 23-25, 2020.

10. **Hee-Gon Kim**, Suhyun Park, Dongnyeong Heo, Stanislav Lange, Heeyoul Choi, Jae-Hyoung Yoo, James Won-Ki Hong, "Graph Neural Network-based Virtual Network Function Deployment Prediction", 16th International Conference on Network and Service Management (CNSM 2020), Virtual Conference, Nov. 2-6, 2020.

11. Namjin Seo, DongNyeong Heo, Jibum Hong, **Heegon Kim**, Jae-Hyoung Yoo, James Won-Ki Hong, and Heeyoul Choi, "Updating VNF Deployment with Scaling Actions Using Reinforcement Algorithms", The 23nd Asia-Pacific Network Operations and Management Symposium (APNOMS 2022), Takamatsu, Japan, Sep. 28-30, 2022.

12. DongNyeong Heo, Doyoung Lee, **Hee-Gon Kim**, Suhyun Park, Heeyoul Choi, "Reinforcement Learning of Graph Neural Networks for Service Function Chaining in Computer Network Management." 2022 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS). IEEE, 2022.

13. Minji Choi, **Hee-Gon Kim**, Jae-Hyoung Yoo, and James Won-Ki Hong, "Net-

work Traffic Prediction and Auto-Scaling of SFC using Temporal Fusion Transformer", 2023 24st Asia-Pacific Network Operations and Management Symposium (APNOMS), Sejong, Korea, Republic of, 2023 **(Best Paper Award)**.

14. Eui-Dong Jeong, **Hee-Gon Kim**, Sukhyun Nam, Jae-Hyoung Yoo, James Won-Ki Hong, "S-Witch: Switch Configuration Assistant with LLM and Prompt Engineering", 1st IEEE/IFIP Workshop on Generative AI for Network Management 2024 (GAIN 2024), Seoul, South Korea, 10 May, 2024. (accepted to appear)

15. **Hee-Gon Kim**, Jae-Hyoung Yoo, James Won-Ki Hong, "AI-based Network Function Virtualization Orchestration", 2024 IEEE/IFIP Network Operations and Management Symposium (NOMS 2024), Seoul, South Korea, 6-10 May, 2024. (accepted to appear)

Domestic Conference Papers

1. **Hee-Gon Kim**, Jae-Hyung Yoo, James Won-Ki Hong, "A Proposal on a Resource Demand Prediction for VNF using Deep Learning", KNOM Conference 2018, Jeju, Korea, May 10-11, 2018, pp. 57-58.

2. Do-Young Lee, **Hee-Gon Kim**, Jae-Hyung Yoo, James Won-Ki Hong, "OpenFlow-based Virtual Gateway for Virtual Networks", KNOM Workshop 2018, Seoul, Korea, Nov. 30, 2018, pp. 16-17.

3. **Hee-Gon Kim**, Seyeon Jeong, Jae-Hyung Yoo, James Won-Ki Hong, "A Study on Machine Learning based VNF Resource Prediction", KNOM Conference 2019, Daegu, Korea, May. 30, 2019, pp. 78-79.

4. **Hee-Gon Kim**, Do-Young Lee, Stanislav Lange, Jae-Hyung Yoo, James Won-Ki Hong, ”The VNF Management System using Machine Learning”, KNOM Conference 2020, On-line KNOM Conference Venue, Korea, May 15, 2020, pp. 64-67.

5. **Hee-Gon Kim**, Jae-Hyung Yoo, James Won-Ki Hong, ”The VNF Management System using Graph Neural Network”, KNOM Conference 2021, On-line KNOM Conference Venue, Korea, April 30, 2021, pp. 1-4.

6. **Hee-Gon Kim**, Jae-Hyung Yoo, James Won-Ki Hong, ”The Hierarchical Graph Learning Model for Network Management”, KNOM Conference 2022, Chuncheon, Korea, May 12-13, 2022. pp. 26-28.

7. **Hee-Gon Kim**, Jae-Hyung Yoo, James Won-Ki Hong, ”AI-based Network Function Virtualization Orchestration”, KNOM Conference 2023, Jeju, Korea, May 18-19, 2023. pp. 58-61.

International Patents

1. James Won-Ki Hong, **Heegon Kim**, Doyoung Lee, Jae-Hyoung Yoo, ”Method of Predicting Demand of Virtual Network Function Resources to which Machine Learning is Applied”, Patent No.: 11,341,372 B2 (16/691,505), USA, 2022.5.24(2019.11.21) (Applicant: POSTECH)

2. James Won-Ki Hong, Jae-Hyoung Yoo, **Heegon Kim**, Taewoo Kim, Seungho Ryu, ”Method of Determining Traffic Path, and Electronic Device Performing The Method,” Patent No.: PCT/KR2023/013909, 2023-09-15 (Applicant: SAMSUNG, POSTECH) (Pending)

3. James Won-Ki Hong, Jae-Hyoung Yoo, **Heegon Kim**, ” Method and Apparatus for Learning Network Management Model Using Hierarchical Graph”, Patent No.: PCT/KR2023/003069, 2023-03-07 (Applicant: POSTECH) (Pending)

4. James Won-Ki Hong, Jae-Hyoung Yoo, **Heegon Kim**, ”Method and Apparatus For Graph Neural Network Based Virtual Network Management”, Patent No.: 17/685301, USA, 2022.03.02 (Applicant: POSTECH) (Pending)

Domestic Patents

1. James Won-Ki Hong, **Heegon Kim**, Doyoung Lee, Jae-Hyoung Yoo, ”Method of Predicting Demand of Virtual Network Function Resources to which Machine Learning is Applied”, Patent No.: 10-2366139 (10-2019-0026890), 2022.02.17 (2019.03.08) (Applicant: POSTECH)

2. James Won-Ki Hong, **Heegon Kim**, Jae-Hyoung Yoo, ”Graph Neural Network Based Virtual Neural Network Management”, Patent No.: 10-2022-0026209, 2022.02.28 (Applicant: POSTECH) (Pending)

3. James Won-Ki Hong, Jae-Hyoung Yoo, **Heegon Kim**, Taewoo Kim, Seungho Ryu, ”Method of Determining Traffic Transfer Route and Electronic Device Performing the Method,” Patent No.: 10-2022-0131630, 2022-10-13 (Applicant: SAMSUNG, POSTECH) (Pending)

4. James Won-Ki Hong, Jae-Hyoung Yoo, **Heegon Kim**, ”Method and Apparatus For Graph Neural Network Based Virtual Network Management”, Patent No.: 10-2022-0156009, 2022-11-21 (Applicant: POSTECH) (Pending)

5. James Won-Ki Hong, **Heegon Kim**, Jae-Hyoung Yoo, Seyeon Jeong, ”Dynamic Service Placement using Reinforcement Learning-based Live Migration”, Patent No.: 10-2023-0099690, 2023-07-31 (Applicant: POSTECH) (Pending)

6. James Won-Ki Hong, **Heegon Kim**, Jae-Hyoung Yoo, Seyeon Jeong, ”Deep Reinforce Learning based Service Allocation Method in Multi-edge Computing Environment and Control Apparatus”, Patent No.: 10-2023-0124955, 2023-09-19 (Applicant: POSTECH) (Pending)

Awards

Title	Organizations	Date
Best Paper Award	APNOMS 2023	2023.09
Student Paper Award	APNOMS 2020	2020.09

Personal Experience

Title	Organizations	Date
Publication Co-chair	NOMS 2024	2024.05

Teaching Assistants

Title	Course	Date
Dept. of CSE, POSTECH	POSTECH MOOC:Software Design	2018 Spring, Fall
	CSED353:Computer Network	2018, 2019 Spring
	CSED702:Internet Traffic Monitoring & Analysis	2023 Spring

References

Prof. James Won-Ki Hong

Department of Computer Science and Engineering

Pohang University of Science and Technology, Pohang, Korea

Prof. Jae-Hyoung Yoo

Department of Computer Science and Engineering

Pohang University of Science and Technology, Pohang, Korea