Master’s Thesis

Machine Learning-based VNF Deployment

Suhyun Park (박 수 현)

Department of Computer Science and Engineering

Pohang University of Science and Technology

2021
기계학습 기반의 VNF 배치

Machine Learning-based VNF Deployment
Machine Learning-based VNF Deployment

by

Suhyun Park
Department of Computer Science and Engineering
Pohang University of Science and Technology

A thesis submitted to the faculty of the Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Master of Science in the Computer Science and Engineering

Pohang, Korea
12. 22. 2020
Approved by
James Won-Ki Hong (Signature)
Academic advisor
Machine Learning-based VNF Deployment

Suhyun Park

The undersigned have examined this thesis and hereby certify that it is worthy of acceptance for a master’s degree from POSTECH

12. 22. 2020

Committee Chair James Won-Ki Hong (Seal)
Member Jae-Hyoung Yoo (Seal)
Member Inseok Hwang (Seal)
ABSTRACT

Network Function Virtualization (NFV) deals with dynamic changes in traffic with appropriate deployment and scaling of Virtualized Network Functions (VNFs). Determining and applying the optimal VNF deployment in consideration of the cost and Quality of Service (QoS) is a complicated and challenging task. In particular, it is necessary to predict the situation at a future point of time when the deployment decision is applied because a certain amount of processing time is required to apply the deployment decision to the actual NFV environment. In this thesis, we randomly generate service requests in a Multi-access Edge Computing (MEC) environment, and then obtain the optimal VNF deployment and Service Function Chaining (SFC) result from an Integer Linear Programming (ILP) solution. We use the simulation data to train a machine learning model that predicts the optimal VNF deployment at a predefined future point of time. The prediction model shows an accuracy of over 90% compared to the optimal ILP solution for the five-minute future time point. In addition, we
apply the trained model to the testbed for the VNF deployment decision against a specific set of service requests and verify its performance by comparing the VNF processing time with the ILP case.
Contents

I. Introduction 1

II. Background and Related Work 4
  2.1 Background ................................. 4
  2.2 Related Work ............................... 6

III. Methodology and Implementation 11
  3.1 Methodology ................................ 11
    3.1.1 Data generation ......................... 12
    3.1.2 Data preprocessing ....................... 14
    3.1.3 Model training ............................ 16
  3.2 Implementation .............................. 17
    3.2.1 Data generation ......................... 17
    3.2.2 Data preprocessing ....................... 20
    3.2.3 Model training ............................ 21
    3.2.4 Module integration ....................... 22

IV. Evaluation 23
  4.1 Results .................................... 23
    4.1.1 Evaluation criteria ...................... 23
    4.1.2 Experimental results .................... 24
4.2 Discussion .......................................................... 27

V. Conclusion ......................................................... 30

Summary (in Korean) ................................................. 32

References ............................................................ 33
List of Tables

3.1 VNF Catalog ................................................................. 12
3.2 An Example of Randomly Generated Service Requests ............ 18
3.3 SFC Catalog ................................................................. 19
3.4 Input Features for the Optimal VNF Deployment .................... 21
4.1 Confusion Matrix for the Model of 5-minute Prediction Horizon with the Best Overall Accuracy ......................... 26
List of Figures

1.1 Traditional Network Environment .................................. 1

2.1 NFV Architecture ..................................................... 4

3.1 Process of Machine Learning Model Development ............... 11

3.2 MEC Environment ..................................................... 12

3.3 Input Data for GNN Encoding ....................................... 15

3.4 Monitoring Time Window (from t-p to t) and Prediction Horizon (from t to t+p) .................................................. 16

3.5 Edge-conditioned filtered Graph Convolutional Neural Network [1] ................................................................. 17

3.6 Simulation Traffic Pattern ............................................. 19

3.7 Integration of VNF Deployment Module with Testbed ............ 22

4.1 An Imaginary Example of VNF Deployment Output Matrix for 4 Servers and 4 VNF Types : ILP solution (left), predicted result (right) ................................................................. 24

4.2 Impact of the Prediction Horizon on the Accuracy of Predicting Deployment .......................................................... 25

4.3 Difference in VNF Deployment Decision Time between ILP and ML according to the Number of Concurrent Service Requests ................................................................. 27
4.4 Difference in VNF Processing Time between ILP and ML for each Service Request
I. Introduction

The demand for high-throughput services on the Internet is increasing. In a traditional network environment, service providers (SPs) have installed many hardware-based middleboxes at a suitable location inside the network and controlled traffic flows to provide various network functions (Fig. 1.1). Network Function Virtualization (NFV) technology is replacing expensive middlebox equipments with Virtualized Network Functions (VNFs) that are implemented as software and run on commodity servers, flexibly coping with the dynamic service request changes [3].

![Figure 1.1: Traditional Network Environment](image)

NFV technology has the advantage of flexibility in dynamically changing network services, although VNF deployment and Service Function Chaining (SFC) tasks are too complicated for the network manager to perform manually in consideration of the operational cost and Quality of Service (QoS). These tasks can be defined as optimization problems, and to solve them, network environment-
related attributes, namely, network topology, server resource allocation, network bandwidth allocation, and catalog for each VNF type and service requests at each time need to be factored in. However, it takes some processing time (VNF deployment time) to deploy VNFs on the server according to the deployment decision, and hence the VNF deployment decision can be sub-optimal at its application time [5].

In the meantime, there are three considerations in specifying the VNF deployment decision problem. First, VNF deployment tasks may have different definitions of optimality depending on the purpose. They generally have two objectives: minimizing the operating costs and maximizing the QoS. In many cases, solutions are being studied to achieve optimization of both the operating costs and the QoS. Second, VNF deployment can cover either scaling or placement depending on the situation: scaling means a decision to add, delete, or maintain the quantity of each type of VNF; placement means a decision including the exact location where each type of VNF will be installed on the network topology. Third, the VNF deployment and the SFC routing path optimizations are closely related to each other. The SFC routing path for each service request should be considered for optimal VNF deployment, and at the same time, the VNF deployment should be decided before choosing specific SFC routing paths. Studies focus on either VNF deployment or SFC routing path decisions, or both at the same time.

In this thesis, we propose a machine learning model to determine the optimal VNF deployment in the Multi-access Edge Computing (MEC) environment. The
MEC is a network architecture concept that enables cloud computing capabilities at the edge network. In this architecture, VNF instances are deployed on edge servers that are closer to users instead of the cloud data center so that network congestion and latency can be reduced. The model has been trained with simulation data from an Integer Linear Programming (ILP) solution involving server resource allocation, network bandwidth allocation, and service requests at each point. ILP calculates the optimal VNF deployment and SFC solution for the given network and service requests. This solution is designed to find the optimal VNF deployment in terms of both costs and QoS. The proposed prediction model uses measurable metrics derived from the simulation data as inputs to predict the optimal VNF deployment in the future. In this study, we consider that VNF deployment includes placement. In addition, we focus on optimizing VNF deployment itself rather than the SFC routing paths.
II. Background and Related Work

2.1 Background

NFV is a network architecture concept that virtualizes hardware-based network functions such as firewalls and load balancers into VNFs that may connect, or chain together, to create various communication services, and it is now being standardized by the European Telecommunication Standardization Institute (ETSI) [2]. Currently, most of the network services are provided by chaining physical network functions (PNFs) implemented on dedicated hardware or VNFs implemented as software components on general-purpose servers (Fig. 2.1).

![NFV Architecture](image)

Figure 2.1: NFV Architecture

The advantage of introducing VNF, which is a network function implemented in software on a general-purpose server, is that the deployment can be dynam-
ically managed and thus cost-effective. The key to NFV management is to efficiently use computing resources while handling incoming requests as much as possible. For this purpose, we need to determine the number of VNF instances for each type to deploy and where to place each VNF instance among the available physical servers with the shortest path for all service requests.

SFC refers to the routing traffic in an ordered list of VNFs that a service request has to go through [4]. It is to construct a chain by selecting each type of VNF, taking into consideration the optimal path in a network environment where multiple VNF instances of the same type are deployed on several physical servers. In this study, the service request corresponding to the traffic demand is a traffic unit specified by arrival time, duration, source/destination node, traffic bandwidth, max latency, penalty, and VNF types to be passed in order. Each SFC is decided for each service request.

Deployment cost is the cost required to install additional VNFs, and energy cost is the cost consumed by the installed VNF. Traffic forwarding cost can be defined as the cost required to pass 1 Mbit data through a single link. The longer the routing path selected is, the higher the cost of traffic forwarding is. To efficiently reduce the traffic forwarding cost, SFC should be configured to allow the shortest path preferentially to traffic.

QoS is a general concept and can be viewed as an indicator of how a customer can receive good service. Service level objective (SLO) is the target value that the service provider wants to achieve in terms of QoS. For each service request or traffic demand, it can be defined differently as end-to-end latency, throughput, or
error rate values. In this study, it is defined as the maximum tolerable end-to-end latency. The penalty is the cost to be paid when the SLO has been violated. For the degree of violation, different rates may be given for each service request, but the same rates are applied for each service request in this study.

2.2 Related Work

VNF instances can be dynamically deployed and controlled in their number and location in an NFV environment. By flexibly adjusting the VNF deployment, administrators can manage network operation in a cost-efficient manner with limited resources such as CPU capacity and network link bandwidth. In addition to scaling, placement needs to be considered to meet QoS requirements such as to reduce delays and maximize cost-efficient resource utilization. This is because the processing time depends on the exact placement of each VNF, even when the numbers of VNF instances for each type are the same.

For the case in which the optimal VNF placement is necessary, first, we can think of a relatively static condition in which the initial VNF placement is the optimization target. Second, in a Multi-access Edge Computing (MEC) environment, users change their locations, and the status of each micro-data center changes. There are previous approaches to these two specific cases. Studies [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19] accommodate cost-efficient VNF placement on virtual infrastructures as an optimization problem through heuristic or approximation algorithms. R. Cohen et al. [7] approached the problem of VNF placement from the perspective of minimizing the cost of deployment
and connection of VNFs, while C. Pham et al. [8] studied it to minimize both energy and traffic costs. Although the problem of VNF placement focused on a relatively static situation, more dynamic and changing aspects of resources in the cloud data centers must be included in this approach. Other studies [3, 20] looked at the provisioning scheme, making placement/ migration decisions as a readjustment problem. M. Ghaznavi et al. [20] formulated an elastic VNF placement problem by considering elasticity overhead and resource consumption. In this readjustment approach, B. Li et al. [3] pointed out that the current responsive readjustment solutions could cause high latencies and endanger the overall QoS to the services of strict requirements such as video conference and voice-over-IP.

As mentioned in the introduction, various studies have been conducted to automate VNF deployment and SFC routing operations in the NFV environment. To automate the VNF deployment operation, we can simply consider developing a program that abides by the decision-making process of human network managers. Recently, attempts have been made to train machine learning models to automate the operations.

M. F. Bari et al. [6] addressed the problem of determining VNF deployment and SFC routing. The problem was defined as the optimization of the network operating costs and utilization while meeting the SLA requirements. ILP for this problem is implemented in CPLEX [21], a mathematical optimization software package. Based on network topology, server resource allocation, network bandwidth allocation, catalog for each VNF type, and service request information at
each time point, the solution yields the optimal VNF deployment and SFC results to minimize VNF deployment cost, energy cost, traffic forwarding cost, and SLA violation penalty. A. Xie et al. [22] considered the chain delay and maximum CPU usage load to determine the edge devices where VNFs are to be installed. They formulated the problem and solved it using ILP. As a result, the maximum load was reduced by 30-50% compared to the existing VNF deployment algorithm. M. Ghaznavi et al. [20] and X. Wang et al. [23] considered server resource capacities and traffic rates between two adjacent VNFs in the service chain to conduct elastic VNF deployment and SFC configuration. The scaling of VNFs can be restricted by resource constraints, and because of unexpected service requests, the scaling of existing VNFs may not result in an optimal solution [3].

X. Zhang et al. [5] pointed out that existing studies on VNF deployment are approaching in a way that responds to service requests that have already arrived. It is argued that the service request needs to be predicted before arrival, thus considering the processing time required to apply the determined deployment. For this purpose, Zhang et al. [5] estimated the upcoming traffic rates and used the result to minimize predefined measures of regret and cost. This study presented a method of proactively planning VNF deployment by combining a learning method and a multi-timescale online optimization algorithm. T. Subramanya et al. [24] proactively predicted the number of needed VNF instances using a variety of machine learning algorithms. In this context, to predict traffic itself, many studies have trained neural networks such as the Bi-Linear Recurrent Neural Network.
or wavelet transform with an artificial neural network [27]. Similarly, B. Li et al. [3] tried to address long setup latency and complex network management problems for provisioning an on-demand Virtual Network Function Service Chain (VNF-SC) in Inter-Data Center Elastic Optical Networks (IDC-EONs). A Deep Learning (DL) model using the Long-Short-Term Memory-based Neural Network (LSTM-NN) [28] was used to accurately predict future VNF-SC requests at a future point. The model was designed to predict multi-dimensional data such as VNF-SC. R. Mijumbi et al. [29] proposed a graph neural network-based algorithm that used the current SFC routing path on the topology graph to predict future resource requirements.

S. Lange et al. [30] predicted the future demand for virtual servers for each VNF type by using randomly generated service request data and the ILP solution proposed in [6]. This study generated service request data in a simulation environment and converted the generated data into traffic data in each time interval as an input value of the ILP solution. In addition, based on the results of this ILP solution, the decision on the increase or decrease in the number of virtual servers for each VNF type at each time point was extracted for labeling. Based on these data, classification models based on machine learning were proposed, and each model received statistical information from the past to the present time as input values and outputted the decision for the quantity of each VNF type in the prediction time horizon (presented as p in Fig. 3.2) as increase/maintain/decrease categories. The proposed model achieved an accuracy of 75% to 80% compared to the ILP result for the optimal quantity change for each VNF type at a time
point after 60 seconds.

This study is an extension of [30, 31] and attempts to improve in two directions. One is to increase the existing prediction horizon from approximately 1 minute to more than 5 minutes, and the other is to specify the location to install for each VNF type in the topology as well as the quantity of each VNF type.
III. Methodology and Implementation

3.1 Methodology

In this section, we explain the process of predicting the optimal VNF deployment and then outline our proposed methodology for randomly generating SFC requests, obtaining an ILP solution, perform data pre-processing, and for training the prediction model as illustrated in Fig. 3.1. In this study, the optimal VNF deployment decision model is designed to predict the optimal VNF deployment after 5 minutes. The model makes the decision based on a service request information in a time window, leading up to the current time point and accordingly changing resource usage of each server and bandwidth usage of each link. The model aims to achieve a high matching score with the ILP solution.

![Figure 3.1: Process of Machine Learning Model Development](image)

Figure 3.1: Process of Machine Learning Model Development
3.1.1 Data generation

**Random SFC request generation** We set MEC as a simulation environment. The topology data includes network topology as a graph where servers are nodes and links are edges, provides CPU resource allocation for each server, and provides network bandwidth allocation for each link. The VNF catalog (Table 3.1) provides the required number of physical CPU cores, delay, and network bandwidth-related capacity for each VNF type. In this study, only leaf nodes on the topology are considered as servers and the rest of the nodes as switches as illustrated in Fig. 3.2. Therefore, only leaf nodes can be the source or destination of service requests and the servers in which VNFs are installed.

![MEC Environment](image)

**Figure 3.2: MEC Environment**

<table>
<thead>
<tr>
<th>Network Function</th>
<th>CPU Required</th>
<th>Processing Capacity</th>
<th>Processing Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firewall</td>
<td>4</td>
<td>900Mbps</td>
<td>45ms</td>
</tr>
<tr>
<td>IDS</td>
<td>8</td>
<td>600Mbps</td>
<td>5ms</td>
</tr>
<tr>
<td>NAT</td>
<td>4</td>
<td>900Mbps</td>
<td>10ms</td>
</tr>
</tbody>
</table>

**Table 3.1: VNF Catalog**
ILP Formulation for VNF Orchestration Problem (VNF-OP)  In this study, we find the optimal VNF deployment and routing path for the randomly generated service requests in each time period by using the ILP solution of [6]. Measurable values (e.g., CPU usage of servers, traffic usage of the links, etc.) are then extracted as input features from the ILP result, and the optimal VNF deployment is obtained as the label data. In [6], the authors aimed to reduce the overall network OPEX and minimize the physical resource fragmentation by provisioning the optimal number of VNFs, placing them in optimal locations and finding the optimal routing path for each traffic request, while meeting the capacity constraints and ensuring that the traffic passes through the proper VNF sequence. Therefore, the cost related to OPEX is defined in four equations.

- **VNF deployment cost**: The cost that is needed to transfer, boot, or attach the VM image

\[ D = \sum_{m \in \mathcal{M} \mid y_m = 1} D^+_p \times q_{mp} \times (y_m - \hat{y}_m) \]

Here, \( D^+_p \) represents the deployment cost of VNF type \( p \). \( q_{mp} \) denotes whether VNF \( m \) is of type \( p \in P \) or not, and \( y_m \) denotes whether VNF \( m \in \mathcal{M} \) is active or not. \( \hat{y}_m \) is the previous status.

- **Energy cost**: The cost of energy consumption for the active servers

\[ E_{\bar{n}} = \sum_{m \in \Omega_{\bar{n}}} y_m \times q_{mp} \times ((e_{\max}^r - e_{\text{idle}}^r) \times \frac{r_c}{r_t} + e_{\text{idle}}^r) \]

Here, \( r_t \) and \( r_c \) denote the total and consumed resources, respectively. \( e_{\text{idle}}^r \) and \( e_{\max}^r \) denote the energy cost in the idle and peak consumption states for resource \( r \), respectively.

- 13 -
• **Traffic forwarding cost**: Leasing cost of transit links and energy consumption of network devices

\[
F = \sum_{t \in T} \sum_{n_1 \in N^t} \sum_{n_2 \in N^t(n_1)} \sum_{\bar{u} \in S} \sum_{\bar{v} \in \eta(\bar{u})} \left( (\omega^{tn_1n_2}_{\bar{u}\bar{v}} - \hat{\omega}^{tn_1n_2}_{\bar{u}\bar{v}}) \times \beta^t \times \sigma \right)
\]

We assume that \( n_2 > n_1 \), and the cost of forwarding 1 Mbit data through one link in the network is \( \sigma \) (in dollars). \( \beta^t \) is the bandwidth demand of traffic. \( \omega^{tn_1n_2}_{\bar{u}\bar{v}} \) denotes whether the link \((n_1, n_2)\) uses a physical link \((\bar{u}, \bar{v})\). \( \hat{\omega}^{tn_1n_2}_{\bar{u}\bar{v}} \) is the previous status.

• **Penalty for SLO violation**: The penalty that must be paid to the customer for SLO violation

\[
P = \sum_{t \in T} \rho^t(\omega^t, \delta^t, \delta^t_a)
\]

Given the policy for determining penalty \( (\omega^t) \), expected propagation delay \( (\delta^t) \) and actual propagation delay \( (\delta^t_a) \) for traffic \( t \) are factors to calculate the penalty for SLO violation.

The objective is to minimize the total network operational cost and resource fragmentation.

\[
\text{minimize}(\alpha D + \beta E + \gamma F + \lambda P + \mu C)
\]

Here, \( C \) is the total cost for resource fragmentation.

### 3.1.2 Data preprocessing

In this study, we train a model that predicts the optimal locations of the required VNF types in 5 minutes by using information such as the network topology, the resource usage of servers, and the traffic usage of links on the network.
Network data can be presented as a graph: the topology (i.e., the relation of nodes and edges), features of each node, and features of each edge. Although the features of each node and edge can be entered independently into the model as input without considering their topology, it is more appropriate to include the relational information of nodes and edges to present network data.

Graph Neural Network (GNN) \cite{32} is a DL-based method that operates within the graph domain and is proposed to collectively aggregate information from graph structure. Therefore, it can model an input and/or output consisting of elements and their dependency. In this thesis, we use the Edge-Conditioned Convolution (ECC) \cite{1}, a GNN algorithm to use the input data represented on the network topology in a graph form, and we define the input data as follows (Fig. 3.3).

![Figure 3.3: Input Data for GNN Encoding](image)

(N: number of nodes, F: node feature dimension, S: edge feature dimension, V: number of VNF types)

- **Node feature**: Server properties including allocated CPU and statistical values of CPU usage represented as $N \times F$ matrix
3.1.3 Model training

We use ECC to deal with our input data in a graph form in which each node and edge has a certain dimension of feature set. Generally, GNNs are used to propagate features across a graph. The authors of reference proposed an ECC to make it more compatible with the common building blocks of current multilayer feed-forward architectures. The ECC approach proposes to condition each filtering weight on the edge features and compute the node as a weighted

Figure 3.4: Monitoring Time Window (from t-p to t) and Prediction Horizon (from t to t+p)

- **Topology**: Network topology represented as $N \times N$ matrix
- **Edge feature**: Link properties including allocated bandwidth and latency and statistical values of traffic usage represented as $N \times N \times S$ array

Statistical values related to CPU usage for the node feature and traffic usage for the edge feature (Table 3.4) were calculated within the monitoring time window (Fig. 3.4). The label data is the future optimal VNF deployment which is represented as an $N \times V$ matrix.
sum of its neighbor nodes (Fig. 3.5). This aggregation approach not only solves the problem of undefined node ordering and varying neighborhood sizes, but also considers structural information. We therefore use the ECC algorithm to handle the graph form of input data and provide structural information of the network topology to the machine learning model.

### 3.2 Implementation

#### 3.2.1 Data generation

**Random SFC request generation** When generating service requests, the overall simulation traffic pattern follows the Abilene network pattern, which is
a known traffic volume pattern in a week (Fig. 3.6). The number of requests per minute, the average duration, and the total time length can be set as input parameters. Then, the arrival interval of the service requests follows the normal distribution to generate a list as in Table 3.2. The number of service requests is controlled according to the simulation traffic pattern. The source and destination nodes are chosen randomly from the node pool. For each service request, one of the three types of SFCs is selected with the probability predefined in the SFC catalog proportion (Table 3.3).

Service request information includes the arrival time and duration of the request, the source node, the destination node, traffic, max latency, and information on the requested order of VNFs (Table 3.2). We assume that SFC requests occur three times per minute, and each service is provided for 900 seconds on average. The amount of traffic over time is consistent with the normalized temporal dynamics of the one-week traffic usage pattern used in [30]. In this way, we generated 22,175 random SFC requests as a simulation for a week.

Table 3.2: An Example of Randomly Generated Service Requests
ILP Formulation for VNF Orchestration Problem (VNF-OP) We set the time interval as the time from the point at which one service request is received to the point at which the next service request is received, and then, we use the active service requests in the time section as the input unit for the ILP program implemented in CPLEX. As a result, the optimal VNF deployment and routing path of the time section is obtained.
3.2.2 Data preprocessing

We use R script to define the input (Fig. 3.3) and the corresponding label dataset; further, three files for each input item and one label data file are extracted as pickle files.

To obtain statistical values as node features and edge features during the time window, we extract data for each time period starting from 5 minutes before the arrival time of each current request to the arrival time of the current request. The minimum, maximum, mean, and slope values (Table 3.4) were obtained from the values. To be precise, each time window consists of the arrival time points of the requests that arrive within the time period. Considering that the average number of requests per minute is 3, the time window data contains 15 sets of usage values on average because we set the time window to 5 minutes.

As mentioned above, the aim of this study is to develop a model that predicts the optimal locations of the required VNF types for longer than 5 minutes. This is a significant increase in the prediction horizon compared to the previous study, so the accuracy is expected to decrease significantly. To understand the effect of the prediction horizon on accuracy, each dataset is created with the values of the prediction horizon from 50 seconds to 300 seconds at 50-second intervals. For the 5-minute prediction horizon, the input data features are extracted from the arrival time of each current request, and the deployment result label data corresponds to the ILP solution at the arrival time of the last request that arrives right before the 5-minute future time point.
Table 3.4: Input Features for the Optimal VNF Deployment

<table>
<thead>
<tr>
<th>Features</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>the number of CPU cores</td>
</tr>
<tr>
<td>cur_node_usage</td>
<td>CPU usage at the current time</td>
</tr>
<tr>
<td>min_node_usage</td>
<td>minimum CPU usage in the time window</td>
</tr>
<tr>
<td>max_node_usage</td>
<td>maximum CPU usage in the time window</td>
</tr>
<tr>
<td>mean_node_usage</td>
<td>mean of CPU usage in the time window</td>
</tr>
<tr>
<td>slope_node_usage</td>
<td>average change rate of CPU usage in the time window</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Features</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>delay</td>
<td>physical delay of the link</td>
</tr>
<tr>
<td>bandwidth</td>
<td>allocated bandwidth of the link</td>
</tr>
<tr>
<td>cur_edge_usage</td>
<td>traffic usage at the current time</td>
</tr>
<tr>
<td>min_edge_usage</td>
<td>minimum traffic usage in the time window</td>
</tr>
<tr>
<td>max_edge_usage</td>
<td>maximum traffic usage in the time window</td>
</tr>
<tr>
<td>mean_edge_usage</td>
<td>mean of traffic usage in the time window</td>
</tr>
<tr>
<td>slope_edge_usage</td>
<td>average change rate of traffic usage in the time window</td>
</tr>
</tbody>
</table>

3.2.3 Model training

Preprocessed input data were fed into the ECC layer. Four feedforward layers are stacked, and each layer has 100, 80, 50, and 20 nodes, respectively. All layers are fully connected, and the Rectified Linear Unit (ReLU) for all hidden layers and Sigmoid for the output layer are used as activation functions. The number of nodes in the output layer of the neural network is $N \times V$ to make the output as the label data format by reshaping. The label data format contains the optimal VNF deployment in 5 minutes after the current time section.

The model was trained in the following environment: Python version 3.6;
TensorFlow version 1.15.0; Keras version 2.3.1. The parameters related to model training are as follows: optimizer = Adam; learning rate = 0.9; batch size = 500.

3.2.4 Module integration

To apply the trained model to the actual testbed, we have implemented the VNF deployment module by loading the model and integrating it to the OpenStack testbed. Simulation data were generated on the testbed topology, and a new model was trained. The module was implemented with RestAPI on Swagger 2.0, an open-source framework. The module consists of the function to retrieve the current VNF deployment status, the resource usage status, the SFC requests list on OpenStack, and the topology information, the function to generate random traffic requests and to deploy VNFs using ILP or Machine Learning (ML) (Fig. 3.7).
IV. Evaluation

4.1 Results

In this section, we present the results of the evaluation of our prediction model with its accuracy. The output of most DL models is usually in a vector form that denotes a value or just a scalar value. On the contrary, in this study, the output is a matrix with the shape of the number of servers × the number of VNF types. With a specific prediction horizon, the dataset could be prepared for supervised learning with the labeled data at the time point. In this unusual setting, we adopted customized evaluation criteria as explained below.

4.1.1 Evaluation criteria

In evaluating the accuracy, it could be possible to simply assume that the correct answer is evaluated only when the predicted values of the entire matrix are exactly the same as those of the label matrix. However, it is highly unlikely that all items match perfectly across the entire matrix because there are as many cases as two to the size of the output matrix.

Therefore, we defined accuracy as the proportion of total matches for each position of the label matrix to measure the closeness of the output with the optimal answer obtained from the ILP result. For example, when the ILP solution and the predicted result are given, the following are the assumptions (Fig. 4.1): one, meaning deployed; zero, meaning not deployed; the accuracy is to be calcu-
lated as $\frac{14}{16}$ for overall, $\frac{5}{6}$ for one, and $\frac{9}{10}$ for zero. This was also applied to the calculation of losses in the training process. The loss is calculated as the sum of the mean squared errors for each location value.

To solve the data imbalance problem between one and zero, the inverse of the frequency of appearance for each value was taken as the weight. By applying a higher weight, we tried to increase the accuracy of the class that appears less.

![Figure 4.1: An Imaginary Example of VNF Deployment Output Matrix for 4 Servers and 4 VNF Types: ILP solution (left), predicted result (right)](image)

4.1.2 Experimental results

To see the effect of the prediction horizon on the accuracy, we generated six datasets with a prediction horizon ranging from 50 seconds to 300 seconds with an interval of 50-seconds and then trained three models with different hyperparameter values for each dataset. Mean accuracy values were obtained from the three models for each dataset. Considering the dynamically changing pattern of SFC requests, we expected the accuracy to be lower when the prediction horizon was longer.
For both 50-second and 100-second prediction horizon datasets, the mean accuracy was approximately 91%, and for all datasets of 150-second, 200-second, and 250-second, the mean accuracy is over 85%. However, the mean accuracy did not gradually decrease. Models trained with the dataset of 100-second prediction horizon show higher prediction accuracy than those of 50-second, and the mean accuracy rises as the prediction horizon increases from 150-seconds to 250-seconds. The mean accuracy of the three models was higher than 84% on average when the prediction horizon was set to 300 seconds, and the model with the highest accuracy recorded an accuracy of over 90% (Fig. 4.2).

Figure 4.2: Impact of the Prediction Horizon on the Accuracy of Predicting Deployment

However, the overall accuracy is not sufficient to evaluate the model. We provide the confusion matrix of the final model (Table 4.1). For the best performing model with the 5-minute prediction horizon, misclassification for the actual one
is very high even when the overall prediction accuracy is higher than 90%. This can be seen as a phenomenon caused by data imbalance during training. To fix this problem, we apply the weight to the loss function according to the frequency of each data appearance. However, this result still needs to be improved.

Table 4.1: Confusion Matrix for the Model of 5-minute Prediction Horizon with the Best Overall Accuracy

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>136,786</td>
<td>6,217</td>
</tr>
<tr>
<td>1</td>
<td>7,486</td>
<td>3,996</td>
</tr>
<tr>
<td><strong>Precision</strong></td>
<td>0.95</td>
<td>0.39</td>
</tr>
</tbody>
</table>

To compare the VNF deployment decision times of ILP and ML, we measured the execution time of each case for 1 to 5 concurrent service requests as input. In the case of ILP, the decision time increased from 0.2 s to 1 s, while the decision time of ML remained almost the same at 0.6 s (Fig. 4.3). We can expect that the decision time of ILP is longer than that of ML as the number of SFC service requests increases while that of ML remains almost the same.

After making an ILP or ML deployment decision for arbitrarily generated service requests in the testbed, we generated HTTP traffic through each of the SFC paths. At this time, the response time for 1,000 requests is measured as the VNF processing time. The optimal solution from ILP includes not only the VNF deployment but also the routing path. However, because the ML decision outputs the result of VNF deployment, the routing path for each service request
is not considered, and VNFs of the same type are grouped and set for load balancing. The ILP decision is processed in the same way as the ML decision. As a result, the VNF deployment from the ML decision achieved a similar level of VNF processing time to that of the optimal deployment from the ILP decision for the 12 individual service requests (Fig. 4.4).

4.2 Discussion

The VNF deployment problem has been addressed in various ways. This study boldly approached the problem in a straightforward manner by using network topology data and such measurable values on the network as server resource usage and link traffic usage values. We present a model that can determine the
optimal VNF deployment at a time point after 5 minutes to the exact location of the server for each VNF type. Once this model is applicable to a real environment, it has a significant value.

However, the limitation of this study is that the model has only been trained and verified with simulation data. Even if the accuracy of the simulation data is high, it cannot guarantee meeting the constraints of the given service requests. For example, according to the results of this model, there is a possibility that a specific VNF type that is essentially required may not be installed at all.

In addition, the reliability of the model is difficult to judge by the overall accuracy of the model as seen in the confusion matrix (Table 4.1). To solve the data imbalance problem, we consider increasing the number of total VNF
deployments by adjusting the data generation conditions. We expect that an even proportion of one and zero could improve the single-class accuracy of both data classes.
V. Conclusion

In this thesis, we intended to predict the optimal VNF deployment in advance by considering VNF deployment time in a given network environment and service requests that change in real-time. For this purpose, we used the optimal VNF deployment result obtained from the ILP solution as the correct answer and trained a machine learning model using the GNN encoding method.

From the results of the experiment, it was observed that an accuracy of more than 90% was achieved for the overall data compared to that achieved for the ILP solution for the 5-minute future time point. However, owing to the data imbalance of one and zero, the accuracy for each of one and zero in the matrix was very low.

We verified the performance of the model in the testbed by comparing the outcome VNF processing time when our model made the VNF deployment decision to that of ILP solution deployment. Our model performs similar to the ILP solution based on the VNF processing time, and hence, this model can be applied to implement the automatic VNF deployment system. Further, it can be used to make VNF deployment decisions with measurable information on the network as input. In addition, it is advantageous to use the ML model, as it takes shorter to complete the VNF deployment decision as the number of concurrent service requests increases.

To improve the reliability of the model, we have two additional considerations
for future work. First, service request information, including the source and destination node identity, may need to be included to provide direct guidance to the model training. Second, we only randomly decided to use the selected statistic values of the monitoring time window, but it is not an optimal way of processing time-series data in terms of providing context information. We therefore plan to improve this study by using a more informative input dataset and a Recurrent Neural Network (RNN)-based machine learning algorithm.

To apply this study to a real NFV environment, the VNF deployment result of the prediction can be used as a baseline. The arrangement of the missing VNF type and SFC routing for each service request need to be additionally conducted. It seems that it can only be used as an auxiliary means to support existing operational methods.
요 약 문

NFV (Network Function Virtualization) 환경은 VNF (Virtualized Network Function)의 적절한 배포 및 확장을 통해 트래픽 상태의 동적 변화를 처리 할 수 있다. 그러나 비용과 서비스 품질 (QoS)을 고려하여 최적의 VNF 배치를 결정하고 적용하는 것은 복잡하고 어려운 작업이다. 특히 실제 NFV 환경에 배치 결정을 적용하는 데는 처리 시간이 걸리기 때문에 배치 결정이 적용될 시점의 상황을 예측할 필요성이 있다.

본 논문에서는 MEC (Multiaccess Edge Computing) 환경에서 무작위로 서비스 요청을 생성 한 다음 ILP (Integer Linear Programming) 솔루션에서 최적의 VNF 배포 및 SFC (Service Function Chaining) 결과를 얻고, 시뮬레이션 데이터를 사용하여 미래 정의된 미래 지점에서 최적의 VNF 배포를 예측하는 기계 학습 모델을 학습시킨다. 예측 모델은 향후 5 분에 대해 ILP 솔루션 대비 90 % 이상의 정확도를 보여준다.

또한 서비스 요청에 대한 VNF 배포 결정을 위해 훈련된 모델을 테스트 베드에 적용하고, 모델을 통해 VNF 배치한 경우와 ILP를 통해 배치한 경우의 VNF 처리 시간을 End-to-end 지연으로 측정 및 비교하여 성능을 검증한다.
References


Acknowledgements

먼저 석사 학위를 성공적으로 이수할 수 있게 저의 결정을 지지해주시고 물심양면으로 도와주신 부모님께 감사의 말씀을 전합니다. 좋은 연구 환경 속에서 다양한 연구를 지도해주시고, 아낌없이 지원해 주신 흥원기 교수님과 유재형 교수님께 깊은 감사의 말씀을 드립니다. 또한 휴직을 도와주신 회사 부서의 팀장님, 원님께도 감사인사를 전합니다.

어찌보면 늦은 나이에 대학원생이라는 새로운 역할에서 예상치 못한 일들도 있었고 지금도 아직 스스로 많이 부족하다고 느끼지만, 되돌아 보니 2년이라는 시간 동안 학업은 물론 다양한 방면에서 많은 값진 경험이 풍부하고 성장을 이룬 것 같은 마음이 듭니다.

석사과정 동안 포항에서의 생활 및 대학원에서의 연구를 시작함에 있어 많은 노하우와 조언을 주신 선배님들, 그리고 같이 연구를 수행하고 함께 활동하며 동고동락한 동기들, 함께한 후배님들 정말 고맙습니다. 모든 선배님들과 동기, 후배님들께 하나하나 감사의 인사를 드리지 못하는 것을 죄송하게 생각합니다. 함께 좋은 시간 보낼 수 있어 감사했습니다, 모두의 앞날에 좋은 일이 가득하기를 기원합니다. 저도 목표하는 바를 이룰 수 있도록 앞으로도 더욱 열심히 하겠습니다.
Curriculum Vitae

Name : Suhyun Park

Research Interest

Software-Defined Networking (SDN); Network Function Virtualization (NFV);
Network Management; Machine Learning (ML)

Education

2004 – 2008    Department of Life Science, Pohang University of Science
                and Technology (B.S.)

2019 – 2021    Department of Computer Science and Engineering, Pohang
                University of Science and Technology (M.S.)
Research/Project Experience

2020. 1. – 2021. 2. 인공지능 기반 가상 네트워크 관리 기술 개발 (Funded by IITP)

Publications: International Conference


2. 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)


4. Heegon Kim, Suhyun Park, Dongnyeong Heo, Stanislav Lange, Heeyoul Choi, Jae-Hyoung Yoo, James Won-Ki Hong, ”Graph Neural Network-based


**Publications: Domestic Journals**


**Publications: Domestic Conference**


