Master's Thesis

A Machine Learning based Anomaly Detection Method for NFV Management

Jibum Hong (홍 지 범)

Department of Computer Science and Engineering

Pohang University of Science and Technology

2020

NFV 환경 관리를 위한 머신러닝 기반의 이상 탐지 방법

A Machine Learning based Anomaly Detection Method for NFV Management

A Machine Learning based Anomaly Detection Method for NFV Management

by

Jibum Hong
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
11. 26. 2019
Approved by
James Won-Ki Hong (Signature)
Academic advisor

A Machine Learning based Anomaly Detection Method for NFV Management

Jibum Hong

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

11. 26. 2019

Committee Chair	James Won-Ki Hong	(Seal)
Member	Jae-Hyoung Yoo	(Seal)
Member	Gwangsun Kim	(Seal)

MCSE 홍 지 범. Jibum Hong
20172480
A Machine Learning based Anomaly Detection Method for NFV Management,
NFV 환경 관리를 위한 머신러닝 기반의 이상 탐지 방법
Department of Computer Science and Engineering , 2020, 47p, Advisor : James Won-Ki Hong. Text in English.

ABSTRACT

Since the concept of Software-Defined Networking (SDN) and Network Function Virtualization (NFV) has been proposed, telcos and service providers have leveraged these concepts to provide their services more efficiently. However, as the virtual network in the data centers becomes more complex, a variety of new network management problems has risen, such as resource allocation, fault management, etc. To deal with these management problems, it is necessary to monitor and analyze resource usage and traffic load of Virtual Network Functions (VNFs) operating on the virtual network. And there have been many attempts to develop technologies that enable network management without human intervention. Many of these attempts based on machine learning techniques are limited to applying arbitrarily chosen machine learning algorithms to detect general resource usage-related anomalies such as high CPU consumption and memory leak. In this thesis, we propose a more targeted approach to detect VNFs' abnormal states

related to SLA violation caused by system resource overload through a comprehensive search for the best machine learning algorithm. We use the datasets collected from the VNFs running on OpenStack environment, and compare the accuracy of the anomaly detection models generated by various machine learning algorithms. Our experimental results show the best model has about 98% accuracy for anomaly detection, and over 95% accuracy in the dataset collected from other VNF scenarios and the environment.

Contents

List of Tables

List of Figures

I. Introduction

Software-Defined Networking (SDN), Network Function Virtualization (NFV), and Network Virtualization (NV) are giving us new ways to design, build and operate networks [1]. With these technologies, telcos and service providers can reduce CAPEX and OPEX by replacing the closed network functions to softwarized Virtual Network Functions (VNFs) [2]. In addition, cloud computing combines these technologies to use computing resources more efficiently in data centers, and provides flexibility and agility for application service deployment and management [3]. For this reason, telcos and service providers recently use SDN/NFV and cloud computing technologies to build a variety of application services, as well as core networks such as virtualized Evolved Packet Core (vEPC) [4] in virtual environments for providing their services. Generally, they build data centers through physical servers, and operate various services on virtual machines (VMs) [5] in NFV environment.

With the wide-spread use of these technologies, the number of VNFs operating on virtual networks has increased dramatically. This complicates virtual networks, causing performance degradation, service failures, and system overload problems. For this reason, the importance of NFV environment management is emphasized. However, unlike managing the existing physical hardware-based network environment, managing virtual networks is more complicated because NFV environment is based on virtual resources. In particular, SDN/NFV makes

it easy for service providers and network administrators to deploy services and configure their networks. However, it is also vulnerable to incorrect decisions and configuration errors, so the management policy based on the wrong decision and misconfiguration can cause unexpected network failures and hardware malfunctions. Therefore, the network administrator has to decide the policies more carefully.

Network administrators can handle these tasks well in small networks, but in the case of large size networks and the networks that have complex dependencies, it requires different approaches. Therefore, recent research on virtual network management using machine learning and deep learning techniques has attracted much attention to solve the above problems. Several attempts have tried to develop technologies that enable the network to understand its status and optimally manage the network without human intervention [6, 7, 8]. One of the main requirements for managing a virtual network is to provide management functions such as detecting abnormal states for resources [9]. With resource management functions, it is able to prevent and detect system overload and faults before serious service failures by monitoring resource usage status.

In this thesis, we present a VNF anomaly detection method based on SLA [10] violation related to resource usage for NFV network management. The proposed method collects the data of VNFs operating in virtual network and detects the abnormal status of VNFs through machine learning algorithms. We use OpenStack testbed to operate VNFs in real-world topology. And we collect data and analyze the optimal model by comparing the accuracy of the models trained

by various machine learning algorithms.

The major contributions of this thesis are as follows.

- Provides comprehensive data generation and collection scenarios for machine learning model training
- Detects VNF's abnormal states considering with the status of the services and networks in the aspects of SLA violation or QoS (Quality of Service)
- Applies the AutoML function to train the models by using various machine learning algorithms and discuss the generated models' generalizability

The remainder of this thesis is organized as follows. Section II provides the background technologies related to our proposed anomaly detection approach and discusses the related issues. We discuss the machine learning algorithms which we used for the proposed anomaly detection method in Section III. Section IV details the proposed anomaly detection methodology and implementation of the method form data collection to data analysis, and Section V evaluates the models by comparing the performance of models trained by machine learning algorithms. Finally, in Section VI, we conclude our thesis and discuss future work.

II. Background and Related Work

2.1 Background

Currently, network management is automated in some processes, but most of them are performed manually by network administrators. However, as network requirements become more diverse and complex, deploying or upgrading their services takes much time to apply, and the demand for expert human resources for network management increases network operation costs. To solve these problems, machine learning, a sub-domain of artificial intelligence, is highly suitable for complex (network) system representation [11].

The network management automation using machine learning generally consists of the processes as shown in Fig. 2.1. First, VNFs operate on the NFV Infrastructure (NFVI) by using its virtual resources, and Analytics monitors VNF resource usage. ETSI NFV Working Group presents representative monitoring data that can be collected and used in an NFV environment (e.g. CPU utilization, memory usage, traffic load, etc.) [12]. Then, the monitoring data is stored in the database. Next, the data is converted into datasets to train the model by using a machine learning algorithm. During training, the machine learning algorithm generates a model which has a specific purpose. After that, the model creates policies on how to operate the virtual network based on the current states. Finally, NFV Management and Orchestrator (MANO) operates and manages the

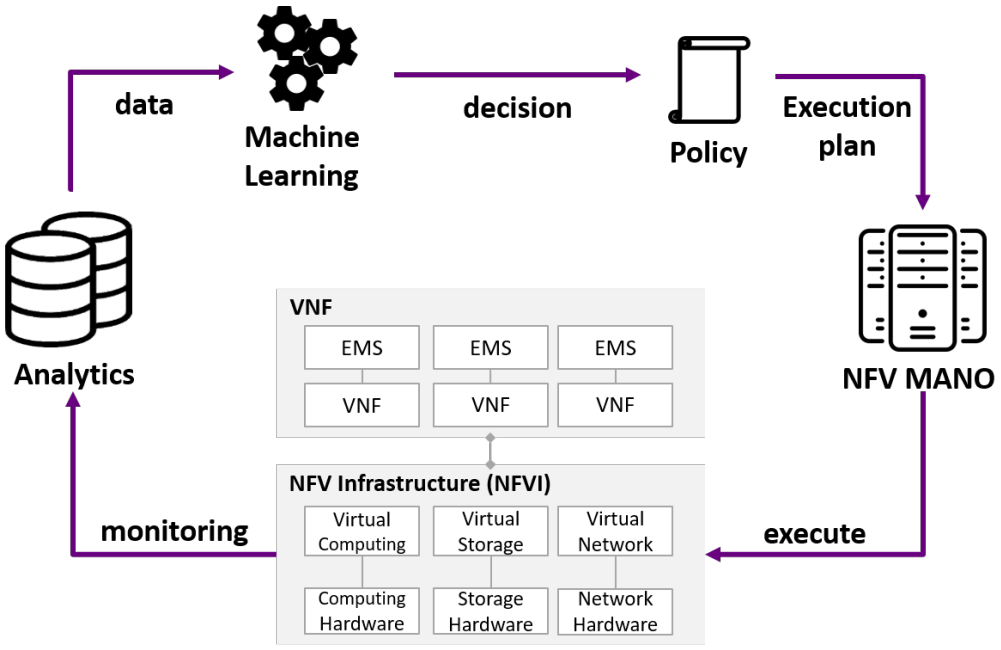


Figure 2.1: Overall process of machine learning based network management

entire NFV environment according to the policies. This machine learning-based network management allows users to optimize the VNF deployment [13] and service chains [14], and predict resource usage [15] in NFV environment.

Besides, among the requirements for network management automation, this work focuses on anomaly detection which is one of the fault management methods for NFV environment. Generally, there exist two types of anomaly detection: 1) system resource anomaly detection and 2) detecting network traffic anomaly detection. System anomaly detection generally monitors the status of VNFs to detect resource bottleneck conditions such as CPU and memory. In network traffic anomaly detection, the model learns the behaviors of network traffic and

detects different patterns than usual, such as the drastic increase in traffic. In this work, we detect abnormal states of VNFs based on resource usage by training monitoring data through machine learning algorithms.

2.2 Related Work

There exist anomaly detection studies in various fields of network management, but most of the studies show many differences depending on the network environments and how to define anomalies. E. Chuah et al. [16] investigated the role of high resource usage on system failures using resource usage monitor and log analysis. In addition, it is very difficult to get datasets related to abnormal situations because they happen rarely and unexpectedly as the term "anomaly" itself implies. So many studies use fault injection techniques to generate the software or hardware faults [17, 18, 19].

To automatically detect anomalies, statistical solutions have been developed. J. Hochenbaum et al. [17] presents statistical measures based on the three-sigma rule, moving averages, and the Seasonal Trend decomposition using Loess (STL) algorithm and compared their efficiencies of detecting anomalies in cloud infrastructure data. The three-sigma rule supposes that the underlying data distribution is normal, and it sets the point which is 3 times of the standard deviation away from the average as a threshold distinguishing anomalies. The moving average is to mitigate the impact of the presence of white noise. STL analysis is being used to exclude the seasonality factor in detecting anomalies in time series data. C. Wang et al. [20] proposed a novel method called EbAT which detects anoma-

lies by analyzing the distribution of arbitrary metrics instead of individual metric threshold, and compared its performance with that of threshold-based method. These statistical approaches might be efficient for automatic anomaly detection when the anomaly is defined by a single value, and the threshold could be clearly set. Otherwise, however, statistical technologies can not classify anomalies which are caused by complex conditions.

As the attempts to adopt the artificial intelligence to network management grow, many studies use machine learning techniques to detect anomalies, especially on the performance of VNFs in NFV environment. C. Sauvanaud et al. [18] proposes the anomaly detection method by classifying the states of VNFs into normal and abnormal, with Random Forest (RF) [21] algorithm which is the supervised machine learning technique. PREPARE system [22] provides automatic performance anomaly prevention for virtualized cloud computing infrastructures by integrating the Markov chain model with the Tree Augmented Naive Bayes (TAN) algorithm. Instead of providing a specific machine learning model, J. Qiu et al. [19] applies Support Vector Machine (SVM) [23], Decision Tree (DT) [24], RF [21], and Neural Network (NN) [25] to detect the performance anomalies, and compare the performance of each model.

There exist several strengths of this study compared to existing ones introduced. Firstly, we collect over 172,000 data for each VNF scenario, while some existing studies use a relatively small number of data for training. Secondly, we generate SLA-related abnormal states by controlling various metrics including CPU utilization, memory usage, disk I/O, and network latency, trying to cover

all possible causes for the anomaly. Lastly, a variety of machine learning algorithms are systematically explored on H_2O [26] framework. In other words, this thesis proposes a method to detect the SLA violation related to system resource and network statistics metrics. Moreover, we take a comprehensive approach with fault injection methods and machine learning techniques which is applicable to commonly used NFV environments.

III. Anomaly Detection using Machine Learning Algorithms

To generate anomaly detection models using machine learning, we use supervised learning-based algorithms among the three machine learning categories (supervised learning, unsupervised learning, and reinforcement learning). Supervised learning returns the corresponding labels when the data is inserted to trained model. Our data is based on the VNFs' resource usage and SLA violation status measured by our monitoring system. In this section, we discuss the three main machine learning algorithms which we adopt in our anomaly detection method.

3.1 Distributed Random Forest

Distributed Random Forest (DRF) [27] is a powerful classification and regression algorithm. DRF generates a forest of classification or regression trees, rather than a single classification or regression tree. Each of these trees is a weak learner built on a subset of rows and columns.

DRF is composed of computing units called "workers" and a coordinating "manager". DRF distributes the dataset between workers so that each worker is assigned to a subset of columns of the dataset. Each worker only needs to read their assigned part of the dataset sequentially. 2 types of workers are assigned for

different operations. The splitters search for optimal splits, and the tree builders build the structure of each decision tree and coordinate the work of the splitter. The manager manages the tree builders. Tree builders and the manager do not have access to the dataset.

DRF builds decision trees each depth level sequentially. As trees of a random forest are independent, DRF trains all the trees in parallel. However, DRF can also train co-dependent sets of trees. In this case, trees cannot be trained in parallel, but the training of each individual tree is still distributed.

3.2 GBM (Gradient Boosting Machine)

While common ensemble techniques like random forests use the average of independently developed models, the boosting methods are based on sequential addition of new models to the ensemble. At each iteration, a base model is trained towards correcting errors of the previous tree. The main idea of this algorithm is to generate the base-learner models to be maximally correlated with the negative gradient of the loss functions, associated with the whole ensemble.

To design a particular GBM [28] for a given task, one has to specify the loss function and the type of base-learner models. The loss function is a form of what to be optimized, and the base models are the unit models which are classified into one of 3 distinct categories: linear models, smooth models, and decision trees.

However, there exist some concerns about overfitting for GBMs and some regularization measures to control it. Firstly, the subsampling procedure requires a parameter which specifies the ratio of the data to be used at each iteration.

Secondly, shrinkage is used for reducing the impact of each additionally fitted base-learner, it reduces the size of incremental steps. This could be regarded as a learning rate. Lastly, early stopping needs to be used with the two mentioned above to avoid overfitting. In addition, the learning procedure is essentially sequential, and this makes GBMs on average slower to learn.

3.3 XGBoost (Extreme Gradient Boosting)

XGBoost [29] is a scalable tree boosting system which runs much faster than existing popular solutions. The most important factors for the scalability include a novel tree learning algorithm for handling sparse data and a weighted quantile sketch procedure which enables handling instance weights in approximate tree learning. By exploiting out-of-core computation, XGBoost enables hundred millions of examples to be processed on a desktop.

XGBoost uses a regularized learning objective to prevent overfitting by adding a term penalizing the complexity of the model. Similarly to GBM, this algorithm adopts shrinkage to reduce the influence of each individual tree and leave space for future trees to improve the model. XGBoost conducts column (feature) subsampling. Using column subsampling is said to prevent overfitting even more so than the traditional row subsampling. The use of column subsampling also speeds up computations of the parallel algorithm.

There exists the exact greedy algorithm which enumerates over all the possible splits on all the features to find the best split. However, to support effective gradient tree boosting, an approximate algorithm first proposes candidate split-

ting points according to percentiles of feature distribution. After that, XGBoost maps the continuous features into buckets split by these candidate points, aggregates the statistics and finds the best solution among proposals based on the aggregated statistics.

3.4 Deep Learning

Unlike the boosting-based tree algorithms as mentioned above, Deep Learning [30] is a machine learning algorithm based on artificial neural networks. Deep Learning utilizes multiple layers to gradually extract key features from a large amount of data or complex data through abstraction. Currently, there exist various Deep Learning algorithms such as Convolutional Neural Network (CNN) [31], Recurrent Neural Network (RNN) [32], and Long-Short Term Memory (LSTM) [33], etc. In our work, we use Feedforward Neural Network (FNN) [34], which is the basic artificial neural network algorithm.

FNN is a type of neural network algorithms in which connections between nodes do not form a cycle. Generally, FNN uses a perceptron structure composed of input layer, hidden layer, and output layer. At this time, Information moves only one direction from the input node to a hidden node through the hidden node because it has no circulation or loop in FNN. The input value is passed to the output through calculation with a series of weights using the activation function. The activation function determines the value, which is the sum of the product of weights of the nodes and the input values. If the value is greater than the threshold value, the activation function has an activation value.

FNN which uses the single-layered perceptron structure is the identically same as the logistic regression model because it can only learn linearly separable patterns in a single hierarchy. However, in the case of a multi-layered structure where each neuron in one layer is connected to a neuron in a subsequent layer, the data can be separated non-linearly.

IV. Methodology and Implementation

4.1 Methodology

In this section, we provide the functional requirements of our anomaly detection method. Fig. 4.1 illustrates an overview of the proposed anomaly detection method. The proposed method consists of 4 main processes, NFVI monitoring, fault injection, preprocessing and training models. We use supervised learning algorithms to learn the relationship between feature data and labeled data. After the main processes, we choose a best model by comparing the performance between the trained models.

4.1.1 NFVI Monitoring

To train the anomaly detection model, we have to monitor the virtual network operating on NFVI. Monitoring functions for the NFVI environment generally consist of monitoring agents, monitoring service, and dashboard.

Monitoring agents collect the resource usage status of each VM running in the virtual network. Monitoring metrics collected by monitoring agents for anomaly detection are composed of 73 metrics which are subdivisions of representative metrics [12] such as CPU utilization, memory usage, and traffic load, etc. Monitoring agents then transfer the data to the monitoring service, then the monitoring service stores the collected data to the time-series database. To provide visibility, the dashboard provides monitoring metrics stored in the database

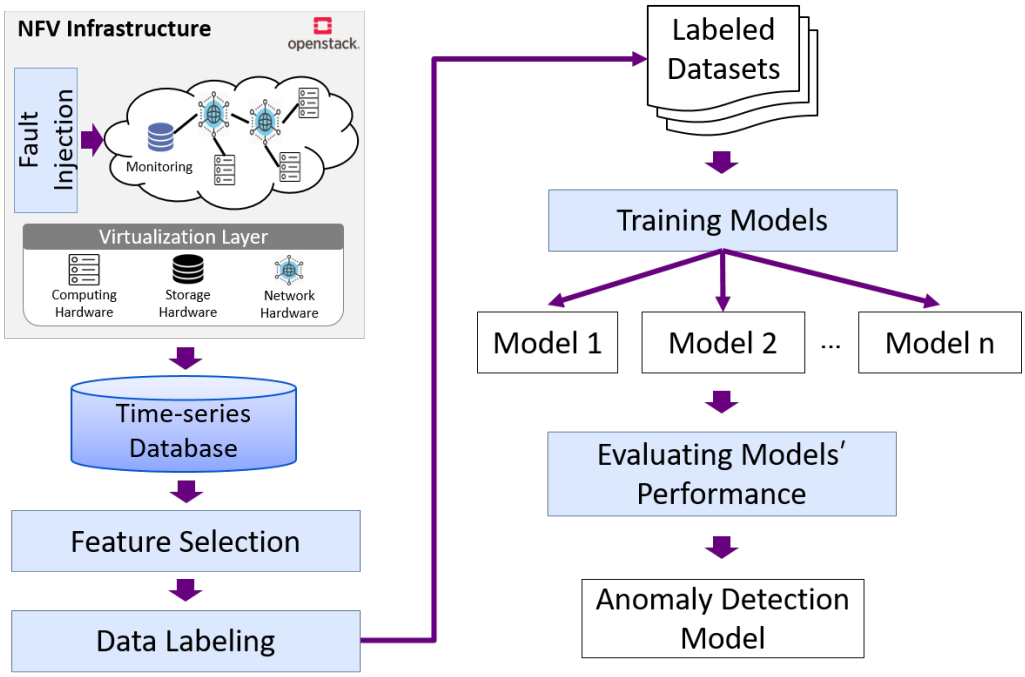


Figure 4.1: Overview of proposed machine learning based anomaly detection method

to the users in the form of graphical visualization. The stored data is transformed into datasets for training after the preprocessing process. We build the monitoring function by using collectd [35] for monitoring agent, InfluxDB [36] for monitoring service and database, and Grafana [37] for the dashboard.

4.1.2 Fault Injection

Because anomalies do not occur frequently on the network, we use fault injection techniques to make the generation of abnormal states precisely controllable. In particular, we emulate the various software or hardware faults which exist in

the system on which the VNFs work. There are two methods to emulate the anomalies, 1) generating the abnormal states to VMs where VNFs operate, and 2) generating heavy workload which does not guarantee the correct service such as sending tremendous traffic.

The first method causes faults directly into the VM where the VNF operates. The fault situations used in this method are considered in terms of CPU utilization, memory usage, disk I/O access, network latency, and network packet loss. The High CPU utilization occurs from a heavy workload of system or from anomalous programs. Lack of memory appears when the allocated memory is accumulated and not freed. Disk access failures occur because of a drastic increase of disk accesses in a short time or lack of disk capacity. Also, network anomalies happen in links between network interfaces. We emulate packet loss and latency increases. We generated faults through stress-ng [38], which is a stress testing tool that causes resource overload, and Linux tc module [39] for network anomaly and traffic control.

The second method causes heavy overload to the network through tremendous traffic. And this method burdens the VNFs because it needs many system resources for handling the packets. Especially, it may cause packet processing delay and packet drop by kernel. We emulate a large amount of traffic to VNFs and a large number of accesses to web servers which pass through the VNFs. We generate the traffic through D-ITG traffic generator [40] to emulate the situation that VNFs can not handle all incoming traffic.

4.1.3 Preprocessing

Preprocessing converts the monitoring data collected through the previous processes into a suitable form for training models. This process consists of feature selection and data labeling (Fig. 4.1). First, feature selection discriminates the metrics which are most relevant to the criteria for distinguishing abnormal states with the 73 metrics collected through monitoring. Then we remove redundant metrics that are intrinsically correlated to each other. In this process, we extracted the 23 features such as for instance information and resource usage of VNFs through feature selection and used the features for training models. Selected features are described in Table 4.1.

Data labeling distinguishes the extracted feature data to normal and abnormal states for training the model through supervised learning-based machine learning algorithms. However, defining abnormal states as the simple situation when the metrics such as CPU utilization temporarily are increased for a very short time causes many false alarms. So we define abnormal states as packet drops occurring inside the VNFs due to the lack of available system resources caused by the fault injection techniques. The packet drop rate is calculated by comparing the number of incoming and outgoing packets processed by the VNFs. We labeled the data as abnormal states when VNFs' packet drops and service request failures occurred over 1% by fault injection, and labeled the rest as normal.

Table 4.1: Selected features for anomaly detection

Features	Description
time	Measurement time
instance	VNF instance name
cpu_idle	CPU - idle time
cpu_interrupt	CPU - interrupt time
cpu_nice	CPU - nice status time
cpu_softirq	CPU - softirq time
cpu_steal	CPU - stolen time by hypervisor
cpu_system	CPU - used by kernel mode
cpu_user	CPU - used by user mode
cpu_wait	CPU - I/O wait time
disk_free	Disk - free space
disk_reserved	Disk - reserved space
disk_used	Disk - used space
disk_read	Disk - I/O read bytes
disk_write	Disk - I/O write bytes
mem_free	Memory - free space
mem_buffered	Memory - buffered space
mem_cached	Memory - cached space
mem_used	Memory - used space
network_rx_bytes	Bandwidth of received packets
network_tx_bytes	Bandwidth of transmitted packets
packet_loss_rate	Network packet loss rate
packet_delay	Network packet latency

4.1.4 Training Models

There are various kinds of machine learning algorithms based on supervised learning to solve classification problems (e.g. SVM, DT, NN, Naive Bayes, and Linear Regression, etc.). To choose the best anomaly detection algorithm, trained models need to be tested with the given dataset for the selection of the best one. In some cases, manual tuning of hyperparameters could be a time-consuming job for the optimized performance. So we chose AutoML function of the H_2O framework [26] which runs most of the algorithms and tunes their hyperparameters to find the best models automatically.

Among tens of models showing the highest performance in validation process, top 4 models are selected. In most of the cases, Models of DRF, GBM, XGBoost, Deep Learning (Feedforward Neural Network) algorithms are included. DRF and GBM are ensemble learning methods, each of them gives a result by combining the outputs from individual trees. GBM builds trees one by one so that each new tree corrects errors made by previously trained trees, while DRF trains each tree independently using a random sample of the data. XGBoost is also based on the gradient boosting, it is generally better than GBM in performance by using a more regularized model to control overfitting. Deep Learning algorithm of H_2O framework is based on a multi-layer feedforward artificial neural network that is trained with stochastic gradient descent using back-propagation. In most cases, default values are provided for many input parameters.

Besides AutoML analysis, we train SVM and DT models because SVM and DT are commonly used classification algorithms for supervised learning. So, we

also compared their performance with that of the models trained by AutoML.

4.2 Implementation

To implement the proposed method, we build the measurement and analysis system to generate the anomaly detection models (Fig. 4.2). The system consists of two parts: data collection and data analysis.

4.2.1 Data Collection

Data collection is to collect the data from VNFs for anomaly detection model training. It is the implementation of NFVI monitoring and fault injection processes in our methodology. We first build VNFs in an OpenStack-based NFVI environment to configure the VNFs' service scenarios. Each scenario emulates the abnormal state of the VNFs through fault injection techniques as well as the normal operation of the VNFs. In each fault injection technique, stress-ng injects faults related to system resource usages used by the VMs, such as high CPU utilization, lack of memory, and disk accesses anomalies. Also, Linux tc generates network latency (packet delay) and packet loss to perform fault injection related to network anomalies and SLA violations.

To monitor the VNFs' normal and abnormal states, we implemented the NFVI monitoring functions. The collectd which is a monitoring daemon agent monitors the status of each VM which VNFs operate (e.g. resource usage, traffic load, etc.). Then, the monitoring data is stored in InfluxDB, a time-series database. Grafana dashboard provides the data stored in the database in the form that users want by sending queries (InfluxQL) to InfluxDB. Finally, we

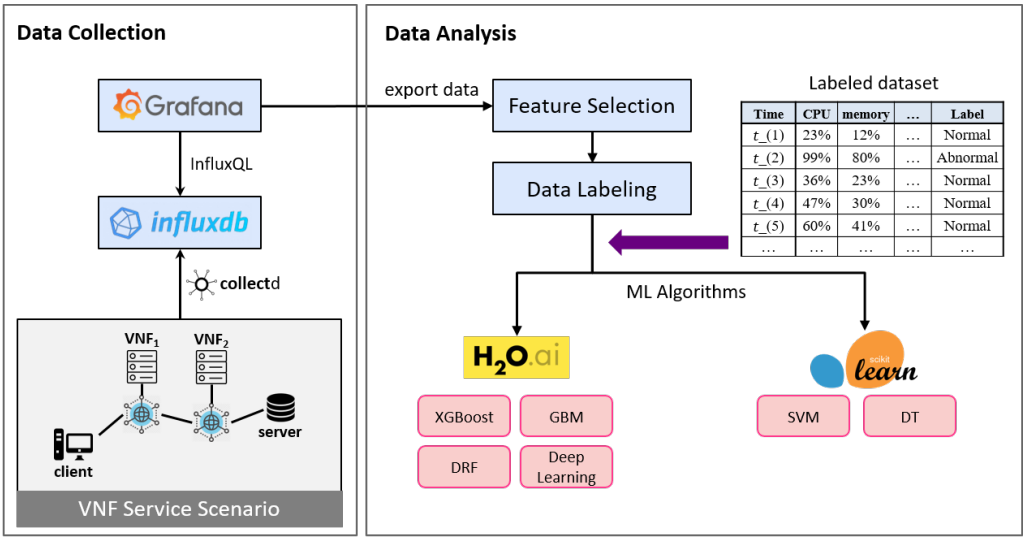


Figure 4.2: Implementation architecture of proposed anomaly detection method

export data for training the anomaly detection models to data analysis.

4.2.2 Data Analysis

Data analysis is the implementation of preprocessing and training models processes. In this part, we train the anomaly detection models through exported data. Among the metrics of VNFs' historical data which is exported from Grafana, we extract 23 features according to the feature selection process (Table I). Then, we label the extracted feature data with normal and abnormal states. Data labeling is performed based on the occurrence of VNFs' abnormal behaviors generated by stress-ng and Linux tc (e.g. high CPU utilization, lack of memory, packet loss, etc).

The labeled dataset created by the preprocessing process learns the relation-

ships between feature data and labeling data through supervised learning-based machine learning algorithms. In this part, we use an H_2O framework with AutoML to train anomaly detection models. AutoML in the H_2O framework supports various machine learning algorithms. Among the many algorithms in H_2O framework with AutoML, we focus on 4 algorithms (XGBoost, GBM, DRF, and Deep Learning) which show the highest performance in our experiments. Also, to compare the models trained by SVM and DT algorithms which are commonly used in supervised learning, we implement the SVM and DT models through the Python scikit-learn [41] library. After each model is generated, we compare the performance of each anomaly detection model and finally select the most suitable model.

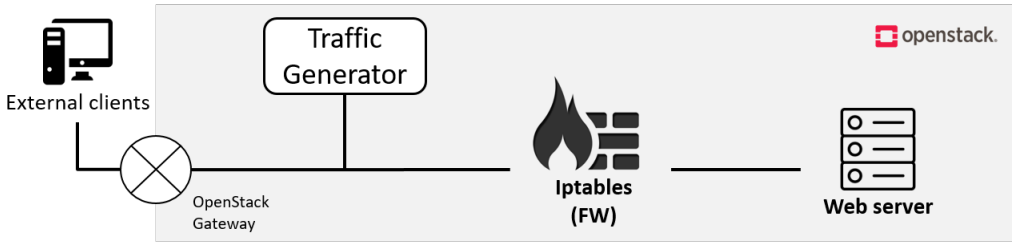
V. Evaluation

5.1 Experimental Setup

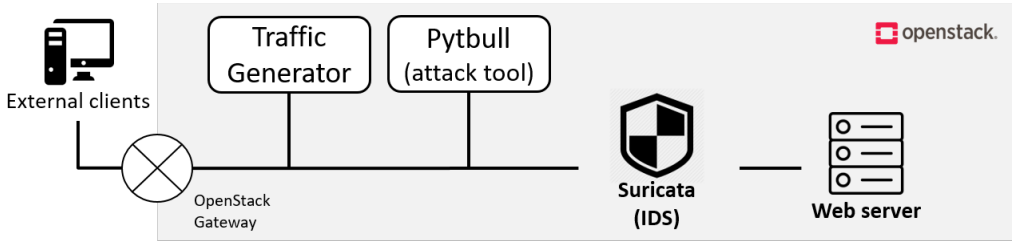
To evaluate the performance of the proposed anomaly detection method, we first set up the experimental testbed environment. We use two hardware servers (Intel Xeon X5650 2.90GHz with 8 cores of 16 threads, 16 GB memory, and 1 Gbps Ethernet) and a network switch to connect the servers. We use an OpenStack (Rocky release) NFVI environment in the testbed.

Fig. 5.1 illustrates our experimental testbed. In the testbed, we construct a virtual network with a monitoring system, VNFs, and web servers. VNFs consist of 3 kinds of open-source VNFs: 1) iptables [42] firewall (FW), 2) Suricata [43] Intrusion Detection System (IDS) / Intrusion Prevention System (IPS), and 3) HAProxy [44] Load Balancer (LB). We allocate 2 vCPUs and 4 GB memory for each VNF to satisfy the VNFs' hardware requirements.

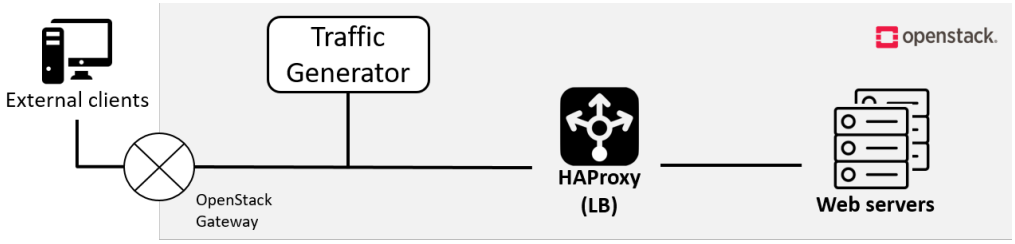
We emulate traffic overload situations through traffic generator and emulate HTTP requests through a web stress tool to generate the client-side HTTP traffic for web servers. Of course, we also install the necessary programs and tools to operate each component.



(a)



(b)



(c)

Figure 5.1: Experimental testbed setup based on OpenStack: firewall (a), intrusion detection system (b), and load balancer (c)

5.2 Experiments

In each scenario, we built VNFs and web servers (Fig. 5.1). The clients outside of the OpenStack testbed send the HTTP requests to web servers in our testbed. For generating traffic overload situations using an internal client (traffic generator in Fig. 5.1), we configured the D-ITG to generate the TCP traffic with the packet size of 1,450 bytes and inter-departure time following constant and Poisson distribution with a mean rate of 100,000 packets per second (PPS). We also emulated the situations when Suricata IDS analyzes 11 kinds of network attacks including DDoS attack generated by pytbull [45] which is a testing tool for IDS/IPS (Fig. 5.1b).

We monitored VNFs' states in every second by generating the abnormal states with fault injection techniques for 48 hours. In each VNF scenario, we collected about 172,000 normal and abnormal data of VNFs. Then, we selected the features from the collected data and labeled the data as mentioned in Section III, and trained the anomaly detection models through various algorithms in AutoML with H_2O framework and scikit-learn library (for training SVM, DT models). To evaluate the trained models' performance more precisely, we divide the labeled data into training data and validation data with a ratio of 75% and 25%. We compare each model's performance with 5-fold cross validation through training data, and validate the models' generalizability through validation data.

5.3 Results

We obtained mean values of accuracy, precision, recall, and F-measure from the validation process. For all models of selected algorithms, labeled data were classified with an accuracy of higher than 0.9. As shown in Table 5.1a, in the firewall-related scenario, XGBoost and GBM showed the best performance, followed by DRF, DT, SVM, and Deep Learning in order. In the case of XGBoost and GBM, the accuracy was about 0.98. The accuracy of DRF was similarly high, while DT, SVM, and Deep Learning performed less than 0.95. Unlike selected models in Table 5.1a, the accuracy of other models based on other machine learning algorithms which trained together through H_2O with AutoML was lower than selected models' accuracy.

We could see more details of cross validation results in the confusion matrix which shows the proportion of correct and incorrect classifications. For firewall experiment, XGBoost model recorded the best performance. As shown in Table 5.1b, out of 17,553 actual abnormal data, 16,097 were correctly classified into abnormal, and the rest 1,436 were incorrectly classified into normal, so the recall of abnormal data became 0.918. In the case of normal data, out of 112,070 actual normal data, 110,803 were correctly classified into normal, and the rest 1,267 were incorrectly classified into abnormal, so the recall of normal data became 0.989. This kind of imbalance was the same for precision. The precision of data classified as abnormal was 0.927, while that of normal was 0.987. It is a usual phenomenon caused by the imbalanced data counts between normal and abnormal classes, and we consider these values of precision and recall for anomalous data as fairly high

Table 5.1: Experimental results of FW-related scenario

(a) Performance comparison of trained anomaly detection models

Algorithm	**Accuracy**	**Precision**	**Recall**	**F-Measure**
SVM	0.9255	0.9225	0.9255	0.9239
DT	0.9468	0.9488	0.9468	0.9375
Deep Learning	0.9295	0.9329	0.9896	0.9604
DRF	0.9646	0.9778	0.9813	0.9795
GBM	0.9794	0.9893	0.9868	0.9880
XGBoost	0.9791	0.9885	0.9873	0.9879

(b) Confusion matrix for best-performing model

Actual	**Predicted**			**Recall**
	Abnormal	Normal	Total	
Abnormal	**16,097**	1,436	17,533	0.918
Normal	1,267	**110,803**	112,070	0.989
Total	17,364	122,239		
Precision	0.927	0.987		

given the imbalanced data counts.

In comparison to firewall experiment, the performance ranking of trained models in IDS-related scenario was on a similar pattern except for SVM model (Table 5.2a). The accuracies of the models except for SVM model were similar which are slightly lower or higher than the models in firewall experiment, XGBoost showed the best performance of about 0.98 in IDS experiment. In contrast, the accuracy of the SVM model in IDS experiment was decreased than the ac-

Table 5.2: Experimental results of IDS-related scenario

(a) Performance comparison of trained anomaly detection models

Algorithm	**Accuracy**	**Precision**	**Recall**	**F-Measure**
SVM	0.9043	0.9038	0.9043	0.9041
DT	0.9455	0.9440	0.9455	0.9446
Deep Learning	0.9305	0.9292	0.9963	0.9602
DRF	0.9684	0.9792	0.9850	0.9821
GBM	0.9821	0.9907	0.9889	0.9898
XGBoost	0.9828	0.9910	0.9894	0.9902

(b) Confusion matrix for best-performing model

Actual	**Predicted**			**Recall**
	Abnormal	Normal	Total	
Abnormal	**14,665**	1,050	15,715	0.933
Normal	1,195	**112,697**	113,892	0.990
Total	15,860	113,747		
Precision	0.925	0.991		

curacy of the SVM model in firewall experiment. As shown in Table 5.2b, the confusion matrix of the cross validation results showed that the results of the IDS experiment were similar to the results of firewall experiment. For abnormal data, precision and recall values were slightly over 0.92, and for normal data, they were around 0.99.

Finally, Table 5.3 shows the experimental results in the LB-related scenario. LB experiment showed similar results to the IDS experiment. XGBoost, GBM,

Table 5.3: Experimental results of LB-related scenario

(a) Performance comparison of trained anomaly detection models

Algorithm	**Accuracy**	**Precision**	**Recall**	**F-Measure**
SVM	0.9222	0.9237	0.9222	0.9229
DT	0.9564	0.9551	0.9564	0.9527
Deep Learning	0.9501	0.9540	0.9919	0.9726
DRF	0.9758	0.9852	0.9876	0.9864
GBM	0.9828	0.9882	0.9925	0.9904
XGBoost	0.9820	0.9880	0.9919	0.9899

(b) Confusion matrix for best-performing model

Actual	**Predicted**			**Recall**
	Abnormal	Normal	Total	
Abnormal	**12,588**	1,392	13,980	0.900
Normal	845	**114,781**	115,626	0.993
Total	13,433	116,173		
Precision	0.937	0.988		

and DRF models showed a very high accuracy of over 0.97. The accuracy of DT and Deep Learning models was lower than that of DRF, GBM, and XGBoost models, and SVM model showed the lowest accuracy. In the confusion matrix in Table 5.3b, precision and recall values were about 0.99 high for normal data similarly to the values of firewall and IDS experiments. For abnormal data, the precision and recall values were lower than the values for normal data.

In other literature mentioned in Section II, while their results showed the

accuracy under 95%, our experimental results showed that the best performing model's accuracy was 98%. This tells us that abnormal states defined by packet drop rate of VNFs are closely related to the abnormal fault injection conditions including significant changes in the measures of other features such as CPU utilization, memory usage, traffic load, network latency, etc.

5.4 Generalizability of Trained Models

In the previous experiments, we trained the model for each VNF scenario to learn about the different characteristics of each VNF. However, one of the most important requirements for applying the trained model by machine learning to the real-world networks is the performance level of the model stably maintained on different instances and environments. To evaluate the generalizability of the trained model and the possibility of model improvement, we validated between the most suitable models of each VNF as follows. We set 25% of labeled datasets which were used in each VNF experiment as the test datasets and individually tested the model with the test datasets of different VNFs.

The experimental results are described as the Receiver Operating Characteristics (ROC) curves (Fig. 5.2). The x- and y- axes represent the true-positive rate (TPR) and false-positive rate (FPR), respectively. Differently colored curves represent the classification results of each model with test datasets from different types of VNFs. Area Under the Curve (AUC) indicates the area below the ROC curve, and it indicates that the larger the AUC value, the better the classification performance of the model for new data. The anomaly detection model for IDS

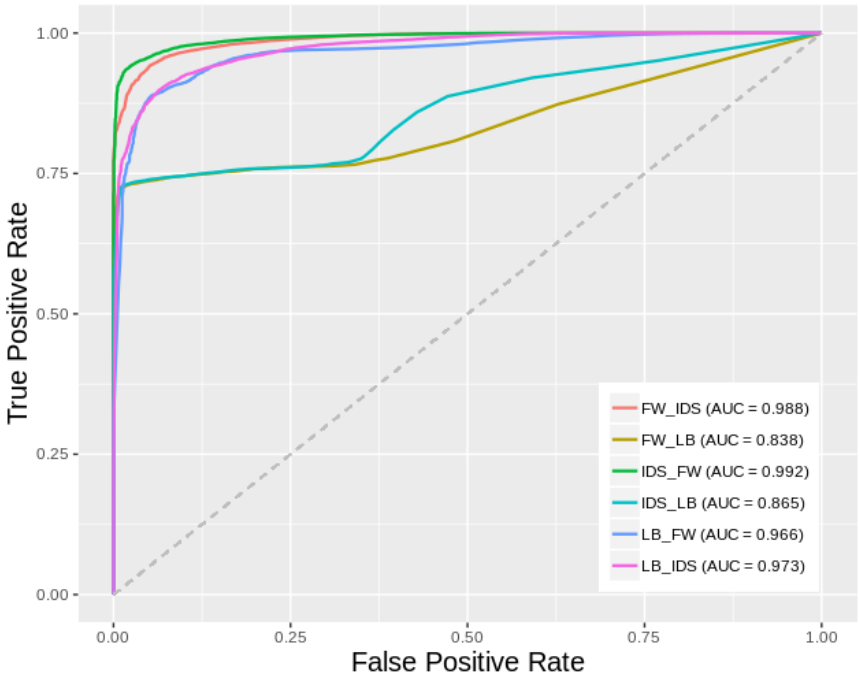


Figure 5.2: ROC curves of anomaly detection models with different test datasets: trained models, test datasets, and AUC values

which was validated by the test dataset of firewall showed the highest classification performance compared to the others. In contrast, the trained models for firewall and IDS which were validated by the test dataset of LB showed a lower accuracy than the other cases. In particular, when the test dataset of the LB was classified by the firewall and IDS models, the results showed worse than other classification results. However, when the trained model for LB was validated by other test datasets, it showed more balanced classification results.

Also, we conducted the experiment with the published test data in [19] with the GBM model which has same hyperparameters in LB-related scenario (Table

Table 5.4: Hyperparameters of GBM model in LB-related scenario

Parameter	**Value**	**Description**
# of trees	125	Number of trees
max_depth	15	Maximum tree depth
min_rows	100	Fewest allowed observation in a leaf
sample_rate	0.8	Row sample rate per tree (from 0.0 to 1.0)
col_sample_rate	0.8	Column sample rate (from 0.0 to 1.0)
col_sample_rate_per_tree	0.8	Column sample rate per tree (from 0.0 to 1.0)
learn_rate	0.1	Learning rate (from 0.0 to 1.0)

5.4). The dataset in [19] is composed of the emulated system monitoring data. The dataset is collected from Clearwater [46] vIMS environment, and labeled to CPU, memory, and disk access overload (the monitoring data is collected form 3 VMs, and each VM has about 1,000 data instances). We use this dataset to detect abnormal behaviors such as CPU, memory, and disk access overload.

We use the up-sampling technique for abnormal data to train the model more accurately because the abnormal data in [19] is small (about 300 data instances in each VM). The experimental results showed 0.98 mean accuracy, and high precision and recall values over 0.95 per each prediction. Analyzing the experimental results through a multi-class confusion matrix (Table 5.5), the classification results (precision and recall values) for normal data were measured about 0.99, which was similar to the values measured in the VNF scenarios on our testbed. The classification results for CPU overload (bottleneck) states were measured to be greater than 0.96 for both precision and recall values, and 0.98

Table 5.5: Experimental results using published dataset on related work's environment

Actual	**Predicted**					**Recall**
	CPU	I/O	Memory	Normal	Total	
CPU	**275**	0	2	7	284	0.968
I/O	3	**416**	0	7	426	0.977
Memory	1	0	**198**	1	200	0.990
Normal	4	8	8	**1,843**	1,863	0.989
Total	283	424	208	1,858		
Precision	0.972	0.981	0.952	0.992		

for detecting disc I/O access overload states. Finally, the memory overload case resulted in a very high recall value of 0.99, while the precision value was about 0.95. This difference is due to the small size of the published dataset, which made the estimated value more sensitive to the small number of errors.

In summary, the performance of the model trained for LB-related scenario was relatively stable when the model was applied to other VNF scenarios or other environments. In our scenario, however, we only verified the case where one VNF operates. Besides, there still exist many other VNFs and various network environments than the ones we evaluated. As a result, the accuracy of anomaly detection could be lower when the trained model is applied to other environments, so further improvements are needed for the application of a generalized model to a wider variety of VNFs in diverse topologies, including service chain scenarios between VNFs.

5.5 Discussion

The attained values from our experiments are relatively higher than any other results of related work while the test environments are different. However, the accuracy measures of anomalous data were relatively lower than those of normal data. Although these results show a little bit lower classification accuracy if the imbalance of the datasets is mitigated, the accuracy still remains high because the recall and precision values of abnormal data are about 0.9 in confusion matrices. To resolve this imbalance, we need to make a balance of the sample counts between normal and abnormal data for the model training. Also, the fault injection we applied could not emulate the entire anomalous situations including application-level failures in the real environment, while it involves system resource overload and SLA violation. To include application-level failures, error messages generated by applications could be used as the measure of labeling for abnormal states.

To prepare for the deployment of trained models to the real operation environment, we evaluated the generalizability of each model. Once we verify the generality of the model's performance, we can deploy the VNFs' anomaly detection model to the NFV management system. Although the models trained with different kinds of VNFs were mostly interchangeable to each other, the model trained with a dataset of LB experiment showed balanced AUC values than other models. Therefore, we expect the model trained by LB-related scenario can perform well to other VNFs or other environments.

For future work, we need to deploy the trained model in our NFV environ-

ment to detect the abnormal states in real-time based on our model. Also, for applying the anomaly detection model on more complex environments, we consider dealing with a wide range of targets such as all VNFs composing service function chains. In addition, anomaly prediction models recognizing anomalous situations in advance are to be developed by extending this work.

VI. Conclusion

6.1 Summary

The current NFV environment faces various demands for efficient management and operation in a fast-changing network. In this thesis, we propose a machine learning-based anomaly detection method for NFV management as a step of network management automation. We also propose a comprehensive methodology for the entire process from data generation and collection to model training and validation. Our trained models are possible to detect the VNFs' abnormal behaviors considering with service and network status through the proposed method.

We configure service scenarios for each VNF in an OpenStack-based testbed with real-world topology, and evaluate the anomaly detection model trained by the proposed method. The results of our experiments showed that the accuracy of the most suitable anomaly detection model is about 98% in web service scenarios. Then, we discuss the generalizability to deploy the trained model to other VNFs or environments. In the generalizability experiments, the results showed that the model generated in the LB-related scenario was able to classify the abnormal states of VNFs with over 95% accuracy in the dataset collected from other VNF scenarios and environment.

6.2 Contributions

With this thesis, we provide the following contributions. First, we provide comprehensive scenarios for data generation and data collection during the machine learning model training processes. Secondly, unlike the existing anomaly detection studies, we detect VNF's abnormal states through generated anomaly detection model considering with the status of the services and networks in the aspects of SLA violation or QoS metrics such as service request failures and packet drop rate. Lastly, we apply the AutoML function to train the models by using various machine learning algorithms, and discuss the generated models' generalizability.

6.3 Future Work

Presently, we implemented the anomaly detection method to find the most suitable model. As future work, we first try to extend the proposed anomaly detection method to provide root cause analysis (RCA), which finds where the faults occurred and what kind of problems happens. Secondly, for more sophisticated validation, we will evaluate our proposed method with more various scenarios such as configuring service chains between multiple VNFs, or environments in vIMS [47] and vEPC [4]. Lastly, we will design and implement the anomaly detectoin system which detects the VNFs' abnormal states in real-time based on the proposed method.

요 약 문

SDN (Software-Defined Networking) 및 NFV (Network Function Virtualization)의 개념이 제안된 이후, 현재 통신 사업자 및 서비스 프로바이더는 SDN/NFV를 활용하여 서비스를 보다 효율적으로 제공하고 있다. 하지만 데이터 센터에서 운용되는 가상 네트워크가 점점 복잡해짐에 따라 리소스 할당, 장애 관리 등과 같은 다양하고 새로운 네트워크 관리 문제가 발생하고 있다. 이러한 관리 문제를 해결하기 위해 가상 네트워크에서 동작하는 VNF (Virtual Network Function)의 리소스 사용 정보 및 네트워크 트래픽 로드를 모니터링하고 분석할 필요성이 있다.

이와 같은 문제를 해결하기 위한 방안으로 최근 사람의 개입없이 네트워크 관리를 가능하게 하는 기술을 개발하려는 많은 시도가 존재한다. 머신러닝 (Machine Learning) 기술을 기반으로 하는 이러한 많은 시도는 임의로 선택된 머신러닝 알고리즘을 적용하여 높은 CPU 사용 상태 및 메모리 누수와 같은 일반적인 리소스 사용과 관련된 이상 상태를 탐지하는 것으로 제한된다.

본 논문에서는 시스템 리소스 과부하로 인해 발생하는 SLA 위반과 관련된 VNF의 비정상 상태를 탐지하는 방법을 제안한다. 제안하는 방법은 가장 높은 분류 정확도를 보이는 모델을 찾기 위해 다양한 머신러닝 알고리즘을 포괄적으로 학습시키는 AutoML을 활용한다. 또한 본 논문에서는 OpenStack 환경에서 구현된 VNF에서 실제로 수집한 데이터셋을 사용하고, 다양한 머신러닝 알고리즘으로 생성된 이상 탐지 모델의 정확성을 비교한다. 실험 결과를 통해 제안하는 방법으로 생성된 이상 탐지 모델이 최고 약 98%의 분류 정확도를 나타낸다. 또한 생성된 최적 모델의

범용성 검증을 위해 다른 VNF 및 네트워크 환경에서 수집한 데이터셋을 통한 분류 실험에서도 95% 이상의 분류 정확도를 나타내는 것을 확인할 수 있다.

References

[1] SDxCentral Staff. Why sdn or nfv now?, Aug. 2013.

[2] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba. On orchestrating virtual network functions. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 50–56, Nov. 2015.

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

[4] J. Kempf, B. Johansson, S. Pettersson, H. Lüning, and T. Nilsson. Moving the mobile evolved packet core to the cloud. In *2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 784–791, Oct. 2012.

[5] Mohammad Masdari, Sayyid Shahab Nabavi, and Vafa Ahmadi. An overview of virtual machine placement schemes in cloud computing. *Journal of Network and Computer Applications*, 66:106–127, 2016.

[6] Raouf Boutaba, Mohammad A. Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano, and Oscar M. Caicedo. A comprehensive survey on machine learning for networking: evolution, applications

and research opportunities. *Journal of Internet Services and Applications*, 9(1), June 2018.

[7] Albert Mestres, Alberto Rodriguez-Natal, Josep Carner, Pere Barlet-Ros, Eduard Alarcón, Marc Solé, Victor Muntés-Mulero, David Meyer, Sharon Barkai, Mike J. Hibbett, Giovani Estrada, Khaldun Ma'ruf, Florin Coras, Vina Ermagan, Hugo Latapie, Chris Cassar, John Evans, Fabio Maino, Jean Walrand, and Albert Cabellos. Knowledge-defined networking. *SIGCOMM Computer Communication Review*, 47(3):2–10, Sep. 2017.

[8] Sara Ayoubi, Noura Limam, Mohammad A Salahuddin, Nashid Shahriar, Raouf Boutaba, Felipe Estrada-Solano, and Oscar M Caicedo. Machine learning for cognitive network management. *IEEE Communications Magazine*, 56(1):158–165, 2018.

[9] IRTF NFV Research Group. Network functions virtualisation – update white paper, 2013. Available at `http://docs.openvswitch.org/en/latest/intro/install/dpdk/`.

[10] Saurabh Kumar Garg, Steve Versteeg, and Rajkumar Buyya. A framework for ranking of cloud computing services. *Future Generation Computer Systems*, 29(4):1012–1023, 2013. Special Section: Utility and Cloud Computing.

[11] D. Rafique and L. Velasco. Machine learning for network automation: Overview, architecture, and applications. *Optical Communications and Networking*, 10(10):D126–D143, Oct. 2018.

[12] ETSI GS NFV-IFA 027. Network functions virtualisation (nfv) release 2; management and orchestration; performance measurements specification. Technical report, 2018. Available at `https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/027/02.04.01_60/gs_NFV-IFA027v020401p.pdf`.

[13] S. Lange, H. Kim, S. Jeong, H. Choi, J. Yoo, and J. W. 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, Sep. 2019.

[14] S. Jeong, H. Kim, J. Yoo, and J. W. Hong. Machine learning based link state aware service function chaining. In *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 1–4, Sep. 2019.

[15] H. Kim, D. Lee, S. Jeong, H. Choi, J. Yoo, and J. W. Hong. Machine learning-based method for prediction of virtual network function resource demands. In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 405–413, June 2019.

[16] E. Chuah, A. Jhumka, S. Narasimhamurthy, J. Hammond, J. C. Browne, and B. Barth. Linking resource usage anomalies with system failures from cluster log data. In *2013 IEEE 32nd International Symposium on Reliable Distributed Systems*, pages 111–120, Sep. 2013.

[17] Jordan Hochenbaum, Owen S. Vallis, and Arun Kejariwal. Automatic anomaly detection in the cloud via statistical learning. *CoRR*, abs/1704.07706, 2017.

[18] C. Sauvanaud, K. Lazri, M. Kaâniche, and K. Kanoun. Anomaly detection and root cause localization in virtual network functions. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pages 196–206, Oct. 2016.

[19] Juan Qiu, Qingfeng Du, Yu He, YiQun Lin, Jiaye Zhu, and Kanglin Yin. Performance anomaly detection models of virtual machines for network function virtualization infrastructure with machine learning. In *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 479–488. Springer International Publishing, 2018.

[20] Chengwei Wang, V. Talwar, K. Schwan, and P. Ranganathan. Online detection of utility cloud anomalies using metric distributions. In *2010 IEEE Network Operations and Management Symposium - NOMS 2010*, pages 96–103, April 2010.

[21] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25(2):197–227, 2016.

[22] Y. Tan, H. Nguyen, Z. Shen, X. Gu, C. Venkatramani, and D. Rajan. Prepare: Predictive performance anomaly prevention for virtualized cloud systems. In *2012 IEEE 32nd International Conference on Distributed Computing Systems*, pages 285–294, June 2012.

[23] Nick Guenther and Matthias Schonlau. Support vector machines. *The Stata Journal*, 16(4):917–937, 2016.

[24] Yan-Yan Song and LU Ying. Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2):130, 2015.

[25] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[26] The H2O.ai team. H2o: Scalable machine learning. Available at `http://www.h2o.ai`.

[27] Mathieu Guillame-Bert and Olivier Teytaud. Exact distributed training: Random forest with billions of examples. *CoRR*, abs/1804.06755, 2018.

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

[29] 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*, KDD '16, pages 785–794. ACM, 2016.

[30] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[32] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[33] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[34] Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Conference on learning theory*, pages 907–940, 2016.

[35] collectd. collectd - the system statistics collection daemon. Available at `https://collectd.org/`.

[36] E. Zimányi S. N. Z. Naqvi, S. Yfantidou. Time series database and influxdb. Technical report, 2017. Available at `http://cs.ulb.ac.be/public/_media/teaching/influxdb_2017.pdf`.

[37] Grafana Labs. Grafana: The open observability platform. Available at `https://grafana.com/`.

[38] Stress-ng. Stress-ng - a tool to load and stress a computer system. Available at `https://manpages.ubuntu.com/manpages/artful/man1/stress-ng.1.html`.

[39] Linux tc. tc - show/manipulate traffic control settings. Available at `https://grafana.com/`.

[40] Alessio Botta, Alberto Dainotti, and Antonio Pescapè. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56:3531—-3547, Oct. 2012.

[41] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, Oct. 2011.

[42] The netfilter.org project. iptables. Available at `http://ipset.netfilter.org/iptables.man.html`.

[43] OISF. Suricata - open source ids/ips/nsm engine. Available at `https://suricata-ids.org/`.

[44] Willy Tarreau. Haproxy - the reliable, high performance tcp/http load balancer. Available at `http://www.haproxy.org/`.

[45] The netfilter.org project. pytbull - open source ids/ips/nsm engine. Available at `http://pytbull.sourceforge.net/`.

[46] Metaswitch Networks. Project clearwater. Available at `https://www.projectclearwater.org/`.

[47] G. Carella, M. Corici, P. Crosta, P. Comi, T. M. Bohnert, A. A. Corici, D. Vingarzan, and T. Magedanz. Cloudified ip multimedia subsystem (ims)

for network function virtualization (nfv)-based architectures. In *2014 IEEE Symposium on Computers and Communications (ISCC)*, volume Workshops, pages 1–6, June 2014.

Acknowledgements

먼저 석사 학위를 성공적으로 이수할 수 있게 어린 시절부터 저를 부살펴 주시면서 물심양면으로 도와주시고 항상 저를 지지해주신 부모님께 감사의 말씀을 전합니다. 또한 좋은 연구 환경 속에서 다양한 연구를 지도해주시고, 학생들의 주변 환경 및 건강, 인격 형성 등에도 많은 조언과 관심을 가져주신 홍원기 교수님과 유재형 교수님께 깊은 감사의 말씀을 드립니다.

포항공과대학교라는 새로운 환경과 학부생에서 벗어나 대학원생이라는 새로운 역할에서 예상치 못한 일들도 있었고 지금도 아직 스스로 많이 부족하다고 느끼지만, 되돌아 보니 3년이라는 시간 동안 학업은 물론 다양한 방면에서 많은 값진 경험을 하고 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 모든 후배님들 정말 고맙습니다. 모든 선배님들과 후배님들께 하나하나 감사의 인사를 드리지 못하는 것을 죄송하게 생각합니다. 함께 좋은 시간 보낼 수 있어 감사했고, 모두의 앞날에 좋은 기운이 함께하기를 기원하겠습니다. 저도 목표하는 바를 이룰 수 있도록 앞으로도 더욱 열심히 하겠습니다.

Curriculum Vitae

Name : Jibum Hong

Research Interest

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

Education

2010 – 2017	Department of Computer Science and Engineering, Hanyang University ERICA (B.S.)
2017 – 2020	Department of Computer Science and Engineering, Pohang University of Science and Technology (M.S.)

Research/Project Experience

2017. 7. – 2020. 2. 글로벌 SDN/NFV 공개 소프트웨어 핵심 모듈/기능 개발 (Funded by IITP)

2018. 7. – 2020. 2. 인공지능 기반 가상 네트워크 관리 기술 개발 (Funded by IITP)

Publications: International Conference

1. **Jibum Hong**, Woojoong Kim, Jae-Hyoung Yoo, James W. Hong, "Design and Implementation of Container-based M-CORD Monitoring System", The 20th Asia-Pacific Network Operations and Management Symposium (APNOMS 2019), Matsue, Japan, Sep. 18-20, 2019.
2. **Jibum Hong**, Seyeon Jeong, Jae-Hyoung Yoo, James Won-Ki Hong, "Design and Implementation of eBPF-based Virtual TAP for Inter-VM Traffic Monitoring", 1st International Workshop on High-Precision Networks Operations and Control (HiPNet 2018), Rome, Italy, Nov. 9, 2018, pp. 402-407.

Publications: Domestic Journals

1. **홍지범**, 정세연, 유재형, 홍원기, "가상 네트워크 트래픽 모니터링을 위한 eBPF 기반 Virtual TAP 설계 및 구현", KNOM Review, Vol. 21, No. 2, Dec. 2018, pp. 26-34.

2. 정세연, **홍지범**, 홍원기, "서버 가상화 환경에서 트래픽 모니터링을 위한 Virtual TAP 설계 ", KNOM Review, Vol. 20, No. 1, Aug. 2017, pp. 1-8.

Publications: Domestic Conference

1. **홍지범**, 김우중, 유재형, 홍원기, "컨테이너 기반 M-CORD 모니터링 시스템 설계 및 구현", KNOM Conference 2019, Daegu, Korea, May 30-31, 2019, pp. 31-32.
2. 박수현, **홍지범**, 유재형, 홍원기, "M-CORD 모니터링 시스템을 이용한 비정상 상태 탐지 연구", KNOM Conference 2019, Daegu, Korea, May 30-31, 2019, pp. 73-74.
3. **홍지범**, 이도영, 최준묵, 유재형, 홍원기, "머신러닝 기반의 VNF Live Migration 기술 연구", KNOM Conference 2018, Jeju, Korea, May 10-11, 2018, pp. 7-8.


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교