Deep Reinforcement Learning based Command Control System for Automating Fault Diagnosis

Hiroshi Yamauchi
Research Institute of Advanced Technology
SoftBank Corp.
hiroshi.yamauchi@g.softbank.co.jp

Tatsuaki Kimura
Faculty of Science and Engineering
Doshisha University
kimura@mail.doshisha.ac.jp

***Abstract*—Due to the recent growth of network services of telecommunication carriers, their communication networks have become increasingly large and complex, and their network operations have also become complicated. For this problem, fault diagnosis and anomaly detection using log data of network devices (e.g., router syslog) have been extensively studied. However, there has been little research on automating the fault diagnosis of failures that are reported by users and whose causes do not appear in logs. In this paper, we propose a deep reinforcement learning (DRL)-based command control system for automating fault diagnosis of communication services. The proposed system first runs a sequence of *device commands* (e.g., show interfaces) for fault diagnosis on network devices, and estimates the fault-type based on the output of the commands using a supervised classifier. The sequence of device commands executed on network devices is optimized by DRL to identify the fault type with the least number of device commands. We evaluate the proposed system using real data obtained from a commercial service network and demonstrate its accuracy and effectiveness.**

***Index Terms*—Deep reinforcement learning, network operation automation, fault diagnosis**

I. Introduction

In recent years, with the rapid expansion of information and communication technologies and services aimed Beyond 5G/6G, the networks of telecommunication carriers are becoming significantly more complex. Ultra-high-speed wireless transmission technologies, realized by Beyond5G/6G and edge computing technology, combined with powered by high-performance computing, enable the construction of a data-driven social environment where virtual and physical spaces, such as digital twins. Further discussions are accelerating toward the effective utilization and progress of information and communication technology.

The demands for telecom carriers as a lifeline are becoming increasingly significant. Concurrently, due to the diverse service development and network expansion, the network becomes large-scale and complex, which makes the network operation tasks, such as fault detection and fault diagnosis, significantly more complicated. Meanwhile, in the realm of network interoperability between enterprises, there is a demand for the realization of scalable and highly reliable operation technologies.

Due to the development of foundational technologies such as network virtualization, microservices, and artificial intelligence operations (AIOps), network operation tasks that have been performed on individual network devices are now possible to be managed, through centralized controller. This allows for the exploration of automating network operations that have been done manually and the flexible interoperability of network operations using external public interfaces between enterprises.

Therefore, there has been active research into technologies for fault diagnosis and anomaly detection using logs of network management systems (NMSs) [1] [2]. Additionally, methods to automate fault diagnosis tasks based on information from fault diagnosis history are being actively studied. Various efforts have been undertaken, such as feature extraction based on syntax discrimination in templates and parameters from syslog [3] [4] [5], and realizing machine learning processing while cross-referencing the supervision information from other systems, such as the trouble-ticket system [6].

However, there are no active methods that aim to automate fault diagnosis tasks, i.e., identifying the root cause of a fault by logging into network devices and executing *device commands* (e.g., show interfaces) to obtain the detailed status of network devices and checking their health. Particularly for faults that cannot be detected by logs, such as Syslog, and are instead identified through user reports, these fault diagnoses can involve complex procedures and substantial management resources, leading to prolonged periods of malfunction. Therefore, automation of this process is crucial.

In this study, we propose a deep reinforcement learning (DRL) [7] based command control system for automating fault diagnosis tasks in network operation. The proposed system automatically identifies the type of failure reported by a customer by executing a minimal series of device commands. When a customer reports the unavailability of a network service, the proposed system first automatically executes a series of device commands on the network devices that comprise the network service. By converting the output of the device commands to a feature vector, the proposed system then estimates the fault type using a supervised classifier. DRL optimizes the sequence of device commands executed on network devices to identify the fault type with the least number of device commands. We evaluate the accuracy of the fault type estimation and the effectiveness of the optimal device command sequence using real data obtained from an actual commercial network.

978-3-903176-59-1

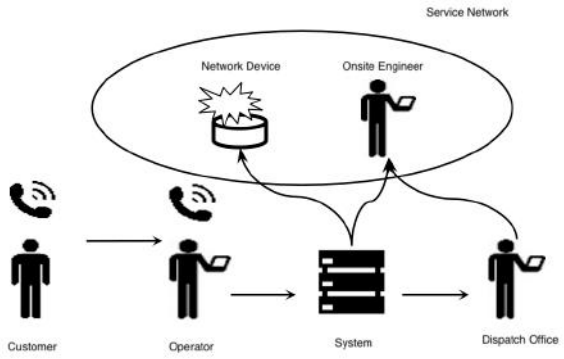

Fig. 1. Trouble-shooting workflow of service failure

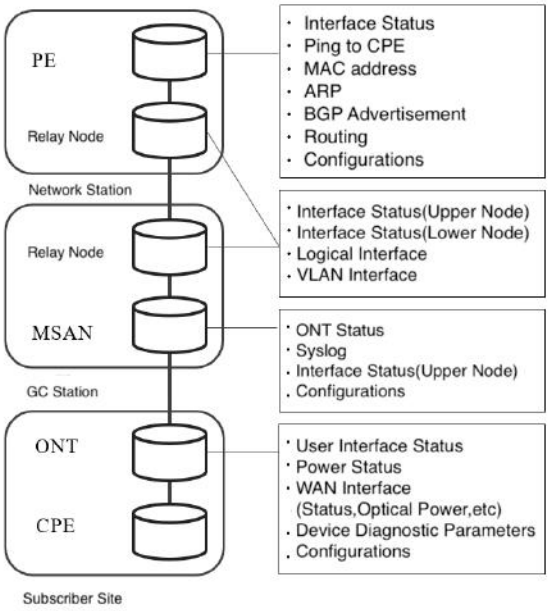

Fig. 2. Confirmation rules using device commands in each network

II. Typical Fault Diagnosis Process in Telecom Carrier Networks

We describe a typical procedure for fault diagnosis in a network operated by a telecom carrier. Fig.1 depicts the overview of this procedure. First, when a user detects a communication service failure, they report it to the customer support office with the service ID assigned to each user's location as the key. The network operator determines the network devices constituting communication service and remotely logs into each device. Then, the network operators execute a set of device commands to obtain the detailed status of the network devices. Typically, network operators have manually constructed *confirmation rules* for device commands to quickly check the health of network devices from the complicated outputs of the device commands. For example, the output of the show interface command shows whether the interface is up or down, and the output of the ping command shows whether the ping rate has exceeded a threshold value. By combining the results of checking the output of device commands with expert knowledge, the network operator can manually identify the root cause of the fault. Then, they arrange workers for failure recovery or instruct on-site recovery work and confirmation of normality between the terminals used by the worker using the failure diagnosis system.

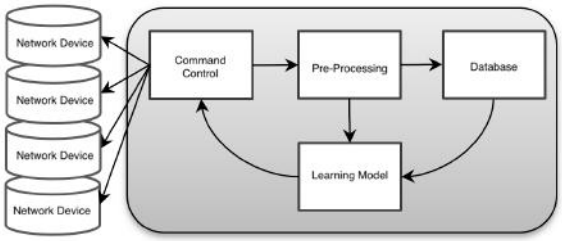

Fig. 3. Automatic device command control system

Fig.2 shows examples of information related to fault diagnosis tasks that can be checked by executing device commands on various network devices and using confirmation rules. Note that the executed device commands for failure diagnosis vary by device and failure type because the information obtained from their output also varies by device command. Therefore, in order to perform fault diagnosis, a suitable set of device commands needs to be selected and executed. Because there are a large number of device commands for each device and their outputs are complex text messages, failure diagnosis is a complicated task dependent on the expert knowledge of network operators.

III. Proposed System

In this section, we explain the proposed deep reinforcement learning-based command control system for automating fault diagnosis.

A. Overview

The overview of the proposed system is depicted in Fig.3. When receiving a report from a user that a communication service is not available, the proposed system first retrieves the detailed information of the network devices that constitute the unavailable services (e.g., IP address, interface information, configuration). The command control unit then logs into each device and executes a set of device commands that are suggested by the learning model unit for checking the details status of devices. The outputs of the device commands are aggregated and converted into a feature vector by the pre-processing unit and stored in the database unit. Stored data is updated by regularly labeling the feature vector data for unknown patterns related to failure events. Then, the proposed system estimates the fault type of the feature vector using a supervised classifier. The optimal sequences of device

commands for each failure type can successfully identify the fault type with the minimum number of device commands based on DRL.

B. Fault Type Estimation by Supervised Classifier

In this section, we describe the fault-type estimation method based on a supervised classifier. First, we explain how to construct the feature vector used for fault-type estimation. It automatically executes a series of commands for each device where a fault is reported. The proposed system automatically matches each pre-defined confirmation rule (see Sec. II) to the output of each device command and classifies each output as 0 (normal) or 1 (abnormal). The proposed system combines these confirmation results of device commands obtained from all network devices related to the failed network service. As the types of device commands executed for each network device vary and each user has different network devices, the dimension of the feature vector is the total number of device commands for all network devices to unify the dimensions of the feature vector for different users. Using the feature vectors of labeled service fault diagnosis datasets as training data, we perform supervised machine learning to estimate the fault type of each feature vector.

C. DRL-based optimal device command sequence generation

In the proposed system, the sequence of device commands that are executed on network devices is optimized by DRL to minimize the number of executed commands. Reinforcement learning [8] is a method of machine learning that observes the current state of a certain environment and decides what actions should be taken to maximize the cumulative reward obtained. DRL is an effective approach for large-scale complex state-action spaces with deep neural networks. The trained model is then sequentially fed with state vectors to obtain an optimal sequence of actions. The value network Q is given by,

$$Q(s_t, a_t) = r_t + \gamma T(s_{t+1}, a_{t+1}) \tag{1}$$

where s_t and a_t denote state and action at time slot t. The value network Q is obtained from the sum of the immediate reward r and the product of the delayed reward estimated by the target function T discount factor γ.

Following the pseudo-code of DRL shown in Fig.4, an agent executes appropriate commands and advances learning while updating the state for all fault cases as episodes repeatedly for iteration time. We prepare the token as a state vector s with the total number of commands as its dimension, action a, and reward r combined with the count of state history. In this experiment, an immediate reward of 100 was given when a command judgment abnormality was detected. While caching tokens up to the buffer size of replay memory D and performing random sampling c, the delayed reward is estimated based on the selected token using the target-network T with weight θ'. Additionally, the weight θ of the value-network Q is updated based on the updated state s, action a, and reward r optimizing by using stochastic gradient descent

Algorithm 1 DQN-based device command sequence generation

Initialize replay memory D.
Initialize value-network Q with random weight θ.
Initialize target-network T with random weight θ'.
Initialize action set A with $zero$.
for iteration=1, 2, ..., N **do**
  for episode=1, 2, ..., M **do**
    if $\epsilon <$ random number between 0 and 1 **then**
      select a random action a_t from A
    else
      select $\mathrm{argmax}\, Q(s_t, a_t, \theta)$
    end if
    Execute action a_t and observe immediate reward r_t and next state s_{t+1}.
    Store transition (s_t, a_t, r_t, s_{t+1}) in D.
    Select randomly samples $c(s_j, a_j, r_j, s_{j+1})$ from D.
    Update A for a_t
    Update weights according to Eq.(2)
  end for
  $T := Q$.
end for

Fig. 4. Pseudo-code of DQN-based device command sequence generation

with respect to the network parameter to minimize the loss function $L(\theta)$ as follows:

$$L(\theta) = [r_t + \gamma \max_{a_{t+1}} T(s_{t+1}, a_{t+1}; \theta') - Q(s_t, a_t; \theta)]^2 \tag{2}$$

Action policy follows the ϵ-greedy policy with probabilities ϵ that decay as the steps progress. In this study, we update the value network Q at each command execution step, but the target network T is updated only after each fault case has been fully explored.

IV. Experiments

In this section, we present the evaluation results of the failure type estimation function using a supervised classifier and the optimal device command sequence using DRL.

A. Dataset

For the evaluation experiment, we use real data obtained from a commercial service network composed of three core networks and one area access network, as shown in Fig.5. The data includes trouble ticket information for each fault that includes the detailed fault diagnosis process accumulated in the fault diagnosis system of this network for the past 6 months. Each log was extracted and labeled by comparing it with the fault reporting ticket. A 121-dimensional feature vector was constructed to represent the types of device commands related to the service accommodation devices that compose

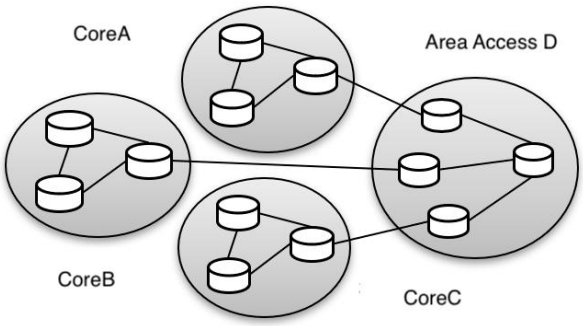

Fig. 5. Service network configuration

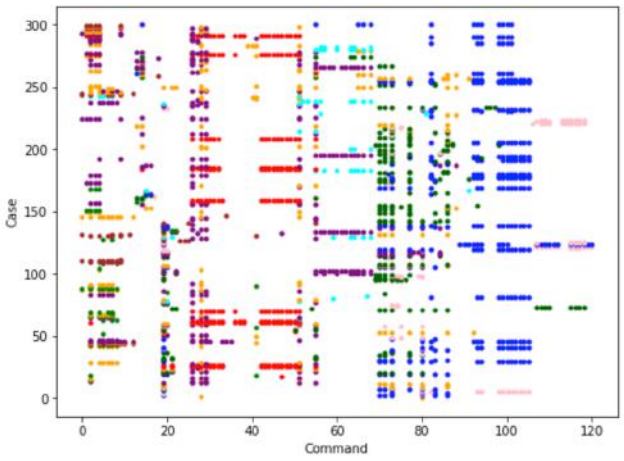

Fig. 6. Service Fault Diagnosis Dataset

TABLE I
FAILURE TYPE

ID	Failure Type	Number of Data
1	Error detection at edge router	28
2	Optical power failure at terminal on user site	35
3	LAN interface failure at terminal on user site	53
4	Optical signal degradation on subscriber segment	15
5	WAN interface failure at terminal on user site	59
6	Configuration error at equipment on station	13
7	Interface error on central relay section	49
8	Packet error on central relay section	33
9	Device error at equipment on station	13

each network. However, each component of the feature vector represents a normal result as 0 and an abnormal result as 1 in the confirmation result of the device command. Of the 5,200 feature vectors obtained, there were 300 unique feature vector patterns, so the 300 fault events corresponding to this feature vector were used as input data in the evaluation experiment.

Fig.6 shows the visualization results of the feature vectors corresponding to the final 300 fault events obtained. The horizontal axis represents the type of device command, the vertical axis corresponds to each fault event, and each point in the graph represents a case where the confirmation result of the device command was determined to be abnormal. The color shows the failure type label. From the figure, it can be seen that there are specific device commands involved in many fault events and that device commands from multiple networks are involved simultaneously, depending on the fault event. Next, Table I shows the failure type labels of communication services defined in advance for the network configuration to be evaluated. Each element of the table shows the label ID, the failure type, and the number of failure events corresponding to each failure type. For each extracted failure event and the corresponding feature vector, the defined label ID was assigned to construct a labeled failure event dataset.

TABLE II
ACCURACY OF FAILURE TYPE ESTIMATION

Model	A	B	C	D	E2E
Multi-layer perceptron	40.4	10.0	10.4	80.3	**82.4**
Perceptron	37.5	10.0	11.0	70.2	74.0
Support vector machine	32.4	30.0	0.0	78.0	79.3
Random forest	40.8	20.0	12.5	80.4	81.3
Logistic regression	34.4	18.7	0.0	80.9	81.0
CART	40.6	12.5	0.0	71.9	77.7
XGBoost	43.8	6.3	0.0	81.8	**82.3**

B. Evaluation of Failure Type Estimation

We conducted an evaluation experiment for fault-type estimation using a labeled fault event dataset and seven types of machine-learning models. In this evaluation experiment, we used seven types of machine-learning models: multi-layer perceptron [9], support vector machine, perceptron, logistic regression, random forest, CART, and XGBoost [10]. The fault-type estimation accuracy was calculated as the proportion of the total fault events that correctly estimated the fault-type labels. The estimation accuracy evaluation was conducted using cross-validation with a split number of 8 for each machine-learning model. Furthermore, in the parameter settings for the multi-layer perceptron, the number of nodes in the intermediate layers was set to 800, 800, and 800 for a three-layer structure, and the activation function used was ReLU (Rectified Linear Unit).

Table II shows the evaluation results for fault-type estimation in each machine-learning model. The evaluation was conducted for the feature vectors corresponding to each network (core A, core B, core C, area access D) and to the end-to-end (E2E) configuration. The following discusses the experimental results. In terms of fault-type estimation accuracy using machine learning models in the end-to-end configuration, the multi-layer perceptron and XGBoost showed high estimation accuracy. The estimation results for end-to-end (E2E) were better than the estimation results for individual network features. This showed that high estimation accuracy can be obtained as a fault diagnosis system by characterizing the states of each device that accommodates communication services across multiple networks when extracting feature vectors from each fault event.

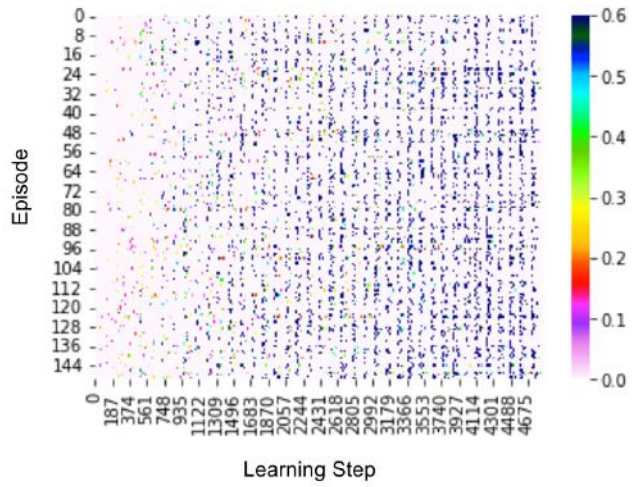

Fig. 7. Transition of Q-value

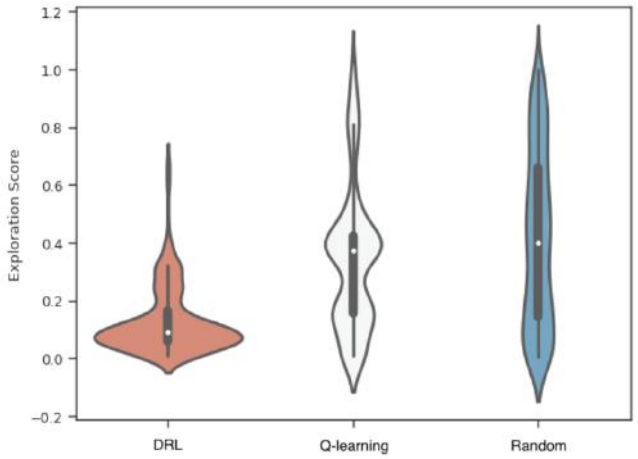

Fig. 8. Efficiency of optimal command control

C. Evaluation of optimal device command sequence

Next, we evaluate the exploration of device command sequence generation based on DRL. Fig. 7 shows the evaluation results for the Q-value transition of DRL. The Q-value increases for each training episode, allowing us to observe the progress of learning. Fig. 8 shows the comparison results of the exploration score of command sequence generation methods using DRL, Q-learning as baseline method, and random selection. Random selection is conducted by executing each step by randomly selecting all commands in such a way that they do not repeat. Each curve represents the distribution of the exploration score that was performed 1000 times for each method, and the good characteristics have a large distribution at smaller values. The exploration score denotes the convergence efficiency to the correct class and is obtained by dividing the number of steps taken to converge to the final class after executing the final command by the total number of commands to normalize. For each step of sequential command execution, the estimation is conducted using the learned model of fault-type estimation created above.

As a result of the experiment, the exploration performance of the DRL model was found to be significantly superior. The learning ability to consider a large past state-action space compared to Q-learning contributes to remarkable performance improvement.

V. Conclusion

In this study, we proposed a system that utilizes deep reinforcement learning to automate device command control related to the diagnosis of communication service failures. We built a dataset for the training data using failure diagnosis logs in commercial service networks and conducted experimental evaluations of the failure type estimation function through the supervised classifier, confirming the superiority of estimation accuracy with multi-layer perceptron and XGBoost. Furthermore, for the optimal search function to diagnose service failures using commands, we evaluated the acquisition of optimal command sequences, confirming the superiority of deep reinforcement learning.

In the future, we will investigate more performance improvements related to learning efficiency and performance enhancement using multi-agent learning and quantum machine learning algorithms on a larger dataset.

References

[1] T. Qiu, Z. Ge, D. Pei, J. Wang, and J. Xu, "What happened in my network: Mining network events from router syslogs," in *Proc. of IMC. ACM*, 2010, pp. 472–484.

[2] C. R. Kalmanek, I. Ge, S. Lee, C. Lund, D. Pei, J. Seidel, J. E. Merwe, J. Yates, "Darkstar: Using exploratory data mining to raise the bar on network reliability and performance," in *Proc. DRCN*, October 2009, pp. 1-10.

[3] W. Meng, Y. Liu, S. Zhang, F. Zaiter, Y. Zhang, Y. Huang, Z. Yu, Y. Zhang, L. Song, M. Zhang, and D. Pei, "Logclass: Anomalous log identification and classification with partial labels," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1870–1884, 2021.

[4] S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao, H. Yang, and Z. Luan, "HitAnomaly: Hierarchical transformers for anomaly detection in system log," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 4, pp. 2064–2076, 2020.

[5] T. Kimura, K. Ishibashi, T. Mori, H. Sawada, T. Toyono, K. Nishimatsu, A. Watanabe, A. Shimoda, and K. Shiomoto, "Spatio-temporal factorization of log data for understanding network events," in *Proc. of IEEE INFOCOM*, 2014, pp. 610–618.

[6] T. Kimura, A. Watanabe, T. Toyono, and K. Ishibashi, "Proactive failure detection learning generation patterns of large-scale network logs," *IEICE Transactions on Communications*, Vol. E102.B (2019), No. 2, pp. 306-316, 2018.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, 518, pp. 529–533, 2015.

[8] R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction*, A Bradford Book, 1998.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations*, Cambridge: MIT Press, 1986, pp. 318-362.

[10] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system", in *Proc. KDD*, August 2016, pp. 785–794.