ComEX

LETTER

Method for network-anomaly detection and failure-scale estimation

Naoya Ogawa[1, 2, a)] **and Ryoichi Kawahara**[2, b)]

Abstract In this study, we propose a novel method for network-anomaly detection and failure-scale estimation using autoencoders, which are a type of neural network. The proposed method first divides the network into several groups. Subsequently, anomalies are detected using an autoencoder for each intergroup traffic, and the failure-scale is estimated from the number of autoencoders that have detected anomalies. We experimentally investigated anomaly detection during communication through a virtual network built using the network emulator Mininet and confirmed that the proposed method can successfully detect anomalies and estimate the failure scale.

Keywords: anomaly detection, failure scale, autoencoder, mininet
Classification: Network system

1. Introduction

Recently, the use of artificial intelligence (AI) in network operations has attracted attention [1]. Although research is progressing in areas such as anomaly detection, root cause analysis, and control recovery, research on estimating failure scales is limited thus far. In several cases, failure scales are estimated manually, and as the sizes and complexities of networks increase, such investigations are expected to become increasingly difficult. To recover rapidly from failures and improve service quality, a method must be established for promptly and automatically estimating the failure scales associated with detected anomalies.

Several related studies on anomaly detection have been conducted. For example, Ikeda et al. [2] reported anomaly detection using an autoencoder (AE), a type of neural network. However, to the best of our knowledge, few studies have been conducted on failure-scale estimation. In [3], anomaly detection in a power-plant system was studied. Although the fault durations were evaluated to determine fault severities, the spatial spread of the faults was not discussed. Hara et al. [4] proposed a method to evaluate the scale of damage in the event of cascading failures associated with traffic volume. However, if we consider various types of network failures besides cascading failures in which the failures are propagated by overloaded traffic, abnormal communications that are unrelated to the traffic volume may occur.

Therefore, in this study, we propose a method for detecting various network anomalies and estimating failure scales using AI. Specifically, the proposed method uses an AE to detect anomalies. The network being monitored is divided into multiple groups, anomalies in intergroup traffic are detected using dedicated AEs, and the failure scale is estimated from the number of AEs in which anomalies are detected.

[1] IDEA Consultants Inc., Setagaya-ku, Tokyo, 154-0012, Japan
[2] Graduate School of Information Networking for Innovation and Design, Toyo University, Kita-ku, Tokyo 115-8650, Japan
[a)] ogw21414@ideacon.co.jp
[b)] ryoichi.kawahara@iniad.org

DOI: 10.23919/comex.2024XBL0028
Received February 16, 2024
Accepted March 1, 2024
Publicized April 8, 2024
Copyedited June 1, 2024

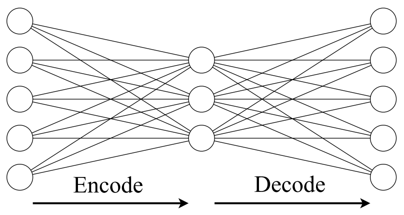

Fig. 1 Image of a basic autoencoder

2. Autoencoder

An AE is an unsupervised learning neural network used in a wide range of fields, including anomaly detection and generation systems. As shown in Fig. 1, an AE can learn the relationships between items of input data by compressing (encoding) dimensions and restoring the input data in its middle and output layers, respectively.

For detecting anomalies, traffic data obtained during normal times are learned such that when normal data are input, the data are restored with a small error. The error used for detecting anomalies is defined as the mean square error (MSE) as follows:

$$MSE(x) = \frac{1}{N}\sum_{i=1}^{N}(\hat{x}_i - x_i)^2. \tag{1}$$

Here, $x = \{x_1, x_2, \ldots x_N\}$ is the input value, $\hat{x} = \{\hat{x}_1, \hat{x}_2, \ldots \hat{x}_N\}$ is the value restored by the AE, and N is the number of dimensions of the input values. During learning, the MSE is optimized to be minimum when inputting the training normal data. A threshold value is set based on the MSE obtained from the normal data, and when the MSE at the time of the test exceeds the threshold value, an anomaly can be detected. This procedure is based on [2].

3. Proposed method

First, we divide the network being monitored into multiple groups, as shown in Fig. 2. Even if network devices have multilayered structures, they are considered to belong to a single group. The finer the grouping, the more accurately the failure scale can be evaluated; however, the computational cost associated with AE learning and anomaly detection may increase; therefore, the granularity of division is determined

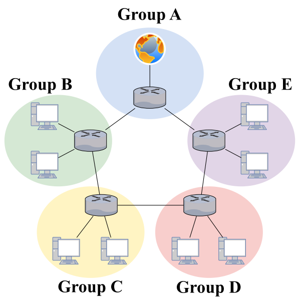

Fig. 2 Network groups

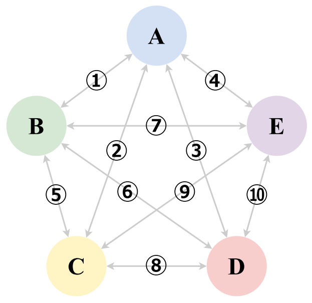

Fig. 3 AEs for each inter-group traffic

depending on the situation and application.

Next, we constructed multiple AEs that can detect anomalies in intergroup traffic. In the example shown in Fig. 2, the network is divided into five groups, therefore, the 10 AEs shown in Fig. 3 are constructed. The evaluation index of the failure scale is determined using Eq. (2) by considering the total number of AEs, M, and the number of AEs, M_A, that detect anomalies.

$$ScaleIndex = \frac{M_A}{M}. \tag{2}$$

This index is 0 if no anomaly has occurred and approaches 1 as the number of AEs detected as anomalies (i.e., the spatial extent of failure) increases; therefore, this index can be used to determine the failure scale. Furthermore, because this method improves the understanding of the traffic between groups in which anomalies occur, the cause of failure may also be investigated using this method.

4. Experiment

4.1 Network topology

Figure 4 shows the virtual network topology built using the Mininet network emulator [5, 6]. In the figure, h1–h9 are hosts and s1–s5 are OpenFlow [7] switches, and Ryu's switching hub [8] is used as the controller. Herein, we assume that h1 is the Internet, h2–h9 are local hosts, and the bandwidth of each link is set to 100 Mbps [1].

4.2 Communications

4.2.1 Internet to local hosts

We used traffic data on April 13, 2023 from Widely Integrate Distribution Environment (WIDE) project [9, 10]. The published data were processed by translating addresses and removing payload data from packets to ensure privacy protection. As traffic cannot flow through the topology in this state, the following preprocessing step was applied:

- Extract packets for eight destination addresses equivalent to the hosts in the topology.
- Convert the destination IP and MAC addresses to match each host in the topology.
- Interpolate removed data from the packet (insert any byte sequence).

[1] Transmission control protocol segmentation offload and generic segmentation offload are enabled on each network interface; therefore, packets larger than the maximum transmission unit may be observed in packet captures.

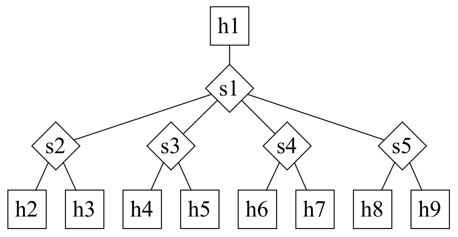

Fig. 4 Network topology

Table I Condition settings used for communication

Source Host	h2–h9
Destination Host	h1 (local hosts to the Internet) h2–h9 (local hosts to local hosts)
Request sending interval	Exponentially distributed random numbers (average 1 s)
Request message length in bytes	Lognormally distributed random numbers (average 1000, standard deviation 1000)
Request failure rate	5 %
Wait time for a response	Uniformly distributed random numbers (100ms – 300ms)
Response message length in bytes	Lognormally distributed random numbers (average 1000, standard deviation 1000)

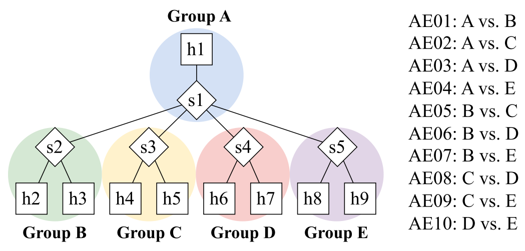

Fig. 5 Topology grouping and constructed AEs

The preprocessed traffic data were replayed from h1, which was assumed to be the Internet, using Tcpreplay version 4.3.4 developed by A. Turner and F. Klassen [11] to ensure communication flow to the topology [2].

4.2.2 Local hosts to the Internet

We sent a request from h2–h9, which are assumed to be local host, to h1, which is assumed to be the Internet host, for transmission control protocol (TCP) communication in which a response was returned from h1 after a short waiting time. Table I lists the corresponding conditions.

4.2.3 Local hosts to local hosts

TCP communication between the local hosts was almost the same as that described in Section 4.2.2; however, the destination of the request was different. Specifically, each time a request was sent, the destination was randomly selected from h2–h9, with uniform probability, excluding the sender.

4.3 Models

4.3.1 Application of the proposed method

As shown in Fig. 5, we divided the topology into five groups and constructed ten AEs that corresponded to the traffic between each group.

[2] Of the replay traffic, we observed 6 Mbps at maximum in s1–s2 on the Mininet. On the other hand, the original size of the corresponding capture data was more than 200 Mbps at maximum. Therefore, packet loss was expected to have occurred in the h1–s1 stage.

Table II Data input to the AEs

Protocol	Item	Units	From h1	From h2–h9
TCP	Number of packets	–	✓	✓
	Total lengths of the packets	bytes	✓	✓
	Average of RTT	ms	✓	✓
	SYN flag content rate	–	✓	✓
	ACK flag content rate	–	✓	✓
	PSH flag content rate	–	✓	✓
	FIN flag content rate	–	✓	✓
	RST flag content rate	–	✓	✓
	Number of source IPs	–	✓	
UDP	Number of packets	–	✓	
	Total lengths of the packets	bytes	✓	
	Number of source IPs	–	✓	
ICMP	Number of packets	–	✓	
	Total lengths of the packets	bytes	✓	
	Number of source IPs	–	✓	

4.3.2 Input data

The measured values obtained from the packet capture were input to the AEs. The measurement period is 2 min, and the items listed in Table II are measured for each source and destination host. Herein, the actual traffic data included packets of the user datagram protocol (UDP) and Internet control message protocol (ICMP) in addition to TCP packets; therefore, the data items differed depending on whether the traffic was sent from h1. Round trip time (RTT) is calculated as the time difference between when a packet with a TCP flag that does not include FIN or RST is obtained and when a response from the destination host is obtained. Because the message lengths fluctuate widely even under normal conditions, a common logarithm was applied to suppress these fluctuations. If the unmodified RTT was input to the AE, the MSE would have been high even for a small delay; therefore, the common logarithm was applied twice. Furthermore, if the total length of the packets within a period was less than 1 byte and the average RTT was less than 10 ms, each value was set to 0; if no TCP packets were observed, the content rate of each TCP flag was set to 1.

4.3.3 Training

Table III presents the design of each AE. Data of normal communications in Section 4.2 were used to train the AEs. As for WIDE traffic data, data collected over one day (672 samples) were used to train the AEs. We trained each AE using certain conditions, such as a val_loss of MSE, a batch_size of 5, 1000 epochs, and a validation_split of 0.2, and the Adam optimizer was used. The input data to the AEs were normalized by dividing each data item by the maximum value among the 672 samples such that the maximum normalized value was 1. In addition, the conditions were adjusted for the anomaly data described later by dividing by the maximum value among the 672 samples. After the training, the val_loss of each AE was less than 0.01, confirming the constructed AEs could restore the data with a small error when normal data were used as input.

4.4 Anomaly detection experiment

4.4.1 MSE threshold

The MSE threshold, MSE_{thres}, used for anomaly detection was determined using Eq. (3) and the MSE, MSE_{train}, of the training data (672 samples) to avoid falsely detecting normal traffic data as anomalies. Table IV lists the MSE threshold of each AE.

$$MSE_{thres} = 2 \times \max MSE_{train}. \tag{3}$$

Table III Design of each AE

AE name	Layer	Dims	Activation	Dropout
AE01–AE04	1	46	–	–
	2	23	Sigmoid	0.2
	3	11	Sigmoid	0.0
	4	23	Sigmoid	0.0
	5	46	Linear	0.0
AE05–AE10	1	64	–	–
	2	32	Sigmoid	0.2
	3	16	Sigmoid	0.0
	4	32	Sigmoid	0.0
	5	64	Linear	0.0

Table IV MSE threshold of each AE

AE	MSE_{thres}	AE	MSE_{thres}
AE01	1.95E-02	AE06	1.75E-02
AE02	3.81E-02	AE07	7.94E-03
AE03	1.08E-01	AE08	1.39E-02
AE04	1.54E-01	AE09	7.61E-03
AE05	1.26E-02	AE10	5.62E-03

Table V List of experiments

Experiment (Expt.)	Description
Expt. 1	50 ms delay on link s1–s2
Expt. 2	500 ms delay on link s1–s2
Expt. 3	10 % packet loss on link s1–s2
Expt. 4	50 % packet loss on link s1–s2
Expt. 5	Limit link s1–s2 bandwidth to 10 Mbps
Expt. 6	Limit link s1–s2 bandwidth to 1 Mbps

Table VI Anomaly detection rate (Case 1: small traffic)

AE	Expt. 1	Expt. 2	Expt. 3	Expt. 4	Expt. 5	Expt. 6
AE01	1.00	1.00	0.93	1.00	0.00	0.00
AE02	0.00	0.00	0.00	0.00	0.00	0.00
AE03	0.00	0.00	0.00	0.00	0.00	0.00
AE04	0.00	0.00	0.00	0.00	0.00	0.00
AE05	1.00	1.00	1.00	1.00	0.00	0.00
AE06	1.00	1.00	1.00	1.00	0.00	0.00
AE07	1.00	1.00	1.00	1.00	0.00	0.00
AE08	0.00	0.00	0.00	1.00	0.00	0.00
AE09	0.00	0.03	0.00	1.00	0.00	0.00
AE10	0.00	0.07	0.00	1.00	0.00	0.00

4.4.2 Overview of the experiments

Based on the MSE of each AE when failures such as delay, packet loss, and bandwidth limitation occurred, we evaluated the anomaly detection and failure-scale estimation capabilities of the proposed method. Table V summarizes the anomaly detection experiments. Two cases were considered in the experiments. A period of low actual traffic flowing from the Internet to h2 and h3 (00:00 to 01:00) represented as Case 1, and a period of high traffic represented as Case 2 (10:00 to 11:00).

4.5 Results of anomaly detection

The percentage of MSE exceeding the threshold obtained using the 1-h dataset (29 samples) used in each experiment was calculated as the anomaly detection rate. Tables VI and VII list the results obtained from Cases 1 and 2, respectively.

The results of Expts. 1 and 2, in which a delay was applied

Table VII Anomaly detection rate (Case 2: large traffic)

AE	Expt. 1	Expt. 2	Expt. 3	Expt. 4	Expt. 5	Expt. 6
AE01	1.00	1.00	1.00	1.00	0.00	0.69
AE02	0.00	0.00	0.00	0.00	0.00	0.00
AE03	0.00	0.00	0.00	0.00	0.00	0.00
AE04	0.00	0.00	0.00	0.00	0.00	0.00
AE05	1.00	1.00	1.00	1.00	0.00	0.69
AE06	1.00	1.00	1.00	1.00	0.00	0.69
AE07	1.00	1.00	1.00	1.00	0.10	0.69
AE08	0.00	0.66	0.00	1.00	0.00	0.00
AE09	0.00	0.72	0.00	1.00	0.00	0.00
AE10	0.00	0.66	0.03	1.00	0.00	0.00

Table VIII Estimated scale indices

Expt.	Average		Standard deviation	
	Case1	Case2	Case1	Case2
Expt. 1	0.40	0.40	0.00	0.00
Expt. 2	0.41	0.60	0.03	0.14
Expt. 3	0.39	0.40	0.03	0.02
Expt. 4	0.70	0.70	0.00	0.00
Expt. 5	0.00	0.01	0.00	0.03
Expt. 6	0.00	0.28	0.00	0.19

to link s1–s2, revealed that the anomaly detection rate corresponding to AE01, AE05, AE06, and AE07 passing through s1–s2 was 1.0. Furthermore, in Case 2 of Expt. 2, the anomaly detection rates corresponding to AE08, AE09, and AE10 were also high. In Case 2, substantial traffic flowed in link s1–s2, and we assumed that a traffic jam occurred, causing a delay that exceeded the set value. AE08, AE09, and AE10 corresponded to communications between local hosts, and the destination was randomly selected each time a request was sent. At this time, if a destination via s1–s2 was selected, a delay occurred, and the number of requests that were sent within the measurement period decreased. Accordingly, an anomaly was considered to have been detected because the number of packets observed decreased even in AE08, AE09, and AE10, which did not travel through s1–s2.

In Expts. 3 and 4, in which packet loss was applied, an anomaly was detected in the traffic passing through s1–s2 by the corresponding AE, and in Expt. 4, in which the packet-loss rate was high, anomalies were detected by AE08, AE09, and AE10 owing to the delay caused by the packet loss.

In Expts. 5 and 6, in which the bandwidth was limited, anomalies were detected when the traffic flow exceeded the bandwidth.

4.6 Result of failure-scale estimation

Table VIII lists the averages and standard deviations of the scale indices obtained from the 29 samples used in each experiment. The scale indices obtained from Expt. 4 were larger than those obtained from Expt. 3. A similar trend was observed in other experiments, and the anomaly strength, including the length of the delay and loss rate, influenced the anomaly detection and failure-scale estimation. Additionally, in Expts. 2 and 6, the scale indices were higher in Case 2 than those in Case 1. This indicated that the amount of traffic affected the results.

5. Conclusion

In this study, we proposed a method for simultaneously detecting network anomalies and estimating the associated failure scales and evaluated the performance of this method through emulated network experiments. Through experiments, we successfully detected anomalies that caused delays, packet loss, and bandwidth limitations and identified that successful anomaly detection and failure-scale estimation depended on the failure strength and the amount of traffic.

Although these experiments were limited to a small-scale topology, we believe that it is possible in principle, this method can be applied to large-scale networks. However, 1–2 min is required for processing the data input from the time of packet capture to reaching the AEs and anomalies being detected, and the high computational cost may be a hindrance to the application of this method to large-scale networks. Although the calculation time largely depends on the performance of the computer being used, the effectiveness of the proposed method when using low-cost traffic data, such as those obtained from flow measurements and packet sampling, must be estimated.

Acknowledgments

We would like to thank Editage (https://www.editage.jp/) for English language editing.

References

[1] R. Kawahara, K. Watanabe, S. Harada, and T. Kawata, "Application of AI to network operation," *IEICE Commun. Soc. Global Newslett.*, vol. 12, no. 1, pp. 29–38, 2020. DOI: 10.1587/bplus.12.29

[2] Y. Ikeda, K. Ishibashi, Y. Nakano, K. Watanabe, and R. Kawahara, "Anomaly detection and interpretation using multimodal autoencoder and sparse optimization," arXiv preprint, 2018. DOI: 10.48550/arXiv.1812.07136

[3] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N.V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 01, pp. 1409–1416, 2019. DOI: 10.1609/aaai.v33i01.33011409

[4] K. Hara and M. Hayashi, "A study of evaluating the impacts of cascading failures in telecommunications networks," IEICE Technical Report, vol. 113, no. 208, pp. 17–22, 2013 (in Japanese).

[5] Mininet Project, "Mininet," https://mininet.org/, accessed Feb. 9, 2024.

[6] D. Oliveira, et al., "Using mininet for emulation and prototyping software-defined networks," 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), pp. 1–6, 2014. DOI: 10.1109/ColComCon.2014.6860404

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008. DOI: 10.1145/1355734.1355746

[8] R. Kudo, R. Kubo, T. Fujita, Y. Agawa, and H. Suzuki, "Ryu SDN framework — open-source SDN platform software," *NTT Technical Review*, vol. 12, no. 8, pp. 18–22, 2014. DOI: 10.53829/ntr201408fa4

[9] WIDE MAWI Working Group, "WIDE traffic archive, Traffic Trace Page," https://mawi.wide.ad.jp/mawi/ditl/ditl2023/, last accessed Dec. 3, 2023.

[10] K. Cho, K. Mitsuya, and A. Kato, "Traffic data repository at WIDE project," USENIX 2000 FREENIX Track, San Diego, CA, June 2000.

[11] AppNeta, "Tcpreplay - Pcap editing and replaying utilitie," https://tcpreplay.appneta.com/, accessed Dec. 3, 2023.