IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 2, 2022, accepted February 17, 2022, date of publication February 25, 2022, date of current version March 9, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3154825

A Mirror Environment to Produce Artificial Intelligence Training Data

DI LI, KAZUAK AKASHI, HARUHISA NOZUE, AND KENICHI TAYAMA
NTT Access Network Service Systems Laboratories, NTT Corporation, Musashino, Tokyo 180-8585, Japan
Corresponding author: Di Li (di.li.us@hco.ntt.co.jp)

ABSTRACT With the increasing maturity of artificial intelligence (AI) technology, business automation technology has also become a trend. Particularly, network operation and maintenance (O&M) is expected to soon become automated and more efficient. However, the automation of O&M is hindered by the lack of network failure data and the cost of collecting data. We thus propose an approach to build a low-cost environment that can produce the same data as the actual production environment and use tools such as chaos engineering to generate training models for fault data. This paper attempts to build the underlying physical network layer using a low-cost single-board computer Raspberry Pi instead of an expensive PC server, while keeping the virtual network layer the same and performing fault simulation, data collection, and AI model training on the constructed virtual network layer. A comparison of the accuracy of the trained AI models verifies the feasibility of replacing the traditional PC server with an inexpensive Raspberry Pi device while keeping the structure and services of the virtual network layer unchanged. Also, a brief comparison with existing techniques is discussed. Our proposed approach solves the problem of insufficient data for AI training while reducing cost and risk.

INDEX TERMS Single-board computers, Raspberry Pi, low cost, network fault, chaos engineering, insufficient data, operation and maintenance.

I. INTRODUCTION

With the rapid development in artificial intelligence (AI) technology, automation is also being used in more areas [1] such as robotic process automation and cyber defense [2]. With the widespread application of AI, the problem of insufficient AI training data has gradually emerged [3]. However, the issue of insufficient training data is undeniable in network operation and maintenance (O&M).

Especially AI, like network fault detection, usually requires many different types of fault data for model training [4], [17]. However, the amount of network fault data that we can collect in real production is minimal. The amount of data we need may be related to the complexity of the problem, the learning algorithm, and the AI model, so the amount of data needed to train an AI model is not easy to accurately estimate. The most common practice is to acquire as much data as possible before processing it [5]. However, most network faults occur very infrequently, so the failures are difficult to reproduce. This leads to training data for network fault AI being more difficult to collect, and the problem of insufficient or nonexistent training data becomes more apparent [6]. There has been research on insufficient AI training data, such as using existing data to augment data, retraining trained AI models, or collecting fault data directly in real production environments. However, none of these approaches has addressed the root of the problem of insufficient AI training data.

Our team put forward an idea to compensate for insufficient AI training data and improve the automation of network (O&M). We can build an environment that can produce and collect the same training data as the actual production environment. Chapter II of this paper introduces the existing methods to solve the problem of insufficient data and its residual problems. Then, Chapter III proposes and explains a solution that can solve the residual problems of existing methods. Next, Chapter IV proves the feasibility of the proposed scheme through a simple experiment, and Chapter V compares the solution with the existing methods. Finally, Chapter VI concludes this paper.

II. BACKGROUND AND PROPOSED APPROACH

This chapter mainly introduces the existing methods to solve the problem of insufficient data and the remaining problems.

The associate editor coordinating the review of this manuscript and approving it for publication was Yiqi Liu.

A. RELATED WORK

Regarding the solution to the problem of insufficient AI training data, there are three main directions of existing approaches: data augmentation, reuse of data or results, and data production.

1) DATA AUGMENTATION

Mikołajczyk and Grochowski proposed a method to increase the original image data by using a series of transformations such as orientation change and color transformation to obtain more training data [7]. This simple and efficient method of increasing training data is widely used in image recognition. Although this method is often used in image processing, care must be taken to preserve the dataset's features while adding data. In addition, this approach requires that the data be available and have the learning features we need before adding data through transformations. However, network fault data is often difficult to obtain and use to identify learning features [8]. Therefore, it does not fundamentally solve the problem of insufficient or nonexistent data.

2) REUSE OF DATA OR RESULTS

We often use methods to compensate for insufficient data in machine learning by reusing AI training data or training results. For example, Zhuang *et al.* proposed using transfer learning [9] when training a neural network (NN), which uses similar trained models or reuses past data to compensate for insufficient training data. Similar methods to transfer learning include meta-learning proposed by Hartmann *et al.* [10] and semi-supervised learning proposed by Shao *et al.* [11]. These methods can alleviate the problem of insufficient training data, but they presuppose the existence of correct datasets or trained AI models. The result is that this reuse approach cannot fundamentally solve the problem of insufficient or nonexistent AI training data.

3) DATA PRODUCTION

Another method is using chaos engineering tools to produce data in the actual production environment [12]. Netflix first proposed chaos engineering to detect potential problems in providing services in advance. They also developed the well-known Chaos Monkey [13], a resiliency tool that helps applications tolerate random instance failures. With the widespread application of chaos engineering technology, many excellent chaos engineering tools have emerged, such as Pumba [14], Gremlin [15], and other tools that can simulate network faults. By using chaos engineering to produce data in a production environment, data can be effectively collected. However, in most cases, this action is accompanied by high risks, such as the system being less able to go back to the normal state than estimated or desired. To reduce these risks, operators must strictly design and control this behavior. In addition, this behavior is generally concentrated when there are relatively few users (like early morning) and requires much pre-planning or real-time observation by professionals.

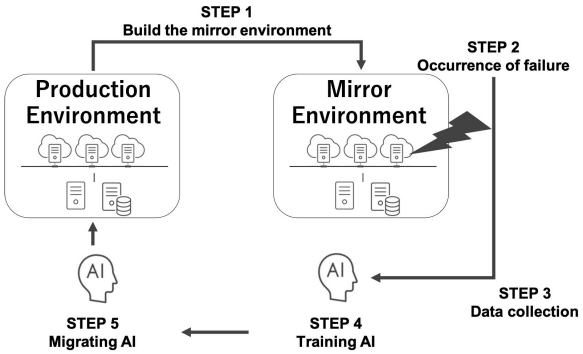


FIGURE 1. AI training using a mirror environment.

This means a reduction in risk accompanied by an increase in late-night labor or labor volume.

B. PROPOSED APPROACH

Data augmentation and reuse methods require correct basic datasets, so these methods cannot solve the problem of insufficient or nonexistent data. Moreover, if the existing dataset is incorrect, a data augmentation or data reuse method may produce results far from the original expectations. Although the data production method using chaos engineering tools in a real production environment can help collect correct data, it has high risks and costs [16]. It also has feasibility problems depending on the industry. For example, it is difficult for providers of network communications services to use. Because the vast infrastructure business always has a physical structure, it is hard to rebuild. Moreover, using chaos engineering tools in a natural production environment has high risks, but the users will not accept any risk of a network fault.

In summary, the existing methods cannot fundamentally solve the problem of insufficient network fault data.

We propose a zero-risk idea that accumulates the same data as that of a production environment. First, a mirror environment of the target production environment is built, and the same data as the actual production environment is collected in this mirror environment. Then, we can use methods such as chaos engineering in this new mirror environment to collect data to compensate for insufficient AI training data.

The concept of data production and collection in the mirror environment is indicated in Figure 1.

First, we need an actual production environment and AI that we want to train. Then, a mirror environment is created (Step 1). We use chaos engineering methods to produce network faults artificially (Step 2) and create and collect data (Steps 3&4) in the mirror environment. Finally, we transfer the AI models trained in the mirror environment to the target production environment (Step 5).

We hope to build a mirror environment for training AI with the same performance as AI trained directly using data from a production environment. This mirror environment

will also use fault injection tools, such as chaos engineering, but it does not present the same risks as the actual production environment. The zero risk of producing data in a mirror environment is because this approach is not performed in an actual production environment. In addition, the proposed approach addresses the legacy issues of the existing techniques mentioned earlier.

III. MIRROR ENVIRONMENT

To implement the approach proposed in Chapter II to solve the problem of insufficient AI training data, this chapter discusses the requirements and functional components of a mirror environment that can generate the same data as the actual target environment.

A. REQUIREMENTS

For the mirror environment to produce and collect AI training data instead of the real production environment and to compensate for insufficient AI training data, the mirror environment needs to satisfy at least four requirements.

1) REPRODUCING THE ENVIRONMENTAL ARCHITECTURE

To collect the same fault data as the target production environment for AI training, the structure of the target production environment needs to be reproduced in the mirror environment.

2) ARTIFICIAL FAULT

To safely collect fault data for AI training, a method is needed to artificially produce the required types of faults (including low frequency and unknown faults).

3) EFFICIENT RECOVERY

To obtain the required AI training data in large quantities in a short time, the recovery time after the occurrence of artificial faults needs to be made as short as possible. Although the required amount of training data depends on the AI, most AI requires hundreds or thousands of training data, so data must be efficiently acquired in a short time for availability.

4) DATA COLLECTION

The proposed approach aims to solve the problem of insufficient data for AI training, so the required data also needs to be collected in a mirror environment. It is also important to collect data in accordance with the type and amount of data we need and to stop collecting data when the required amount of data is reached.

B. CONFIGURATION

A mirror environment that satisfies the above requirements has four main components (Figure 2): the environment reproduction department, the events reproduction department, the recovery department, and the data collection department. The functions of the four departments are as follows

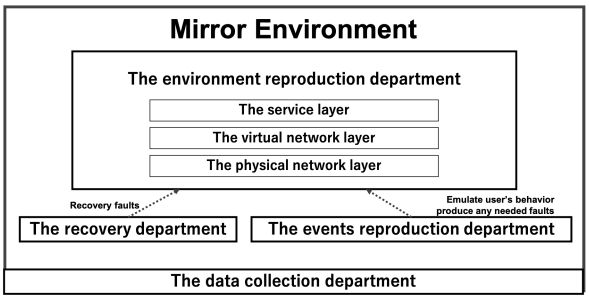


FIGURE 2. Composition of a mirror environment.

1) THE ENVIRONMENT REPRODUCTION DEPARTMENT

This department reproduces the physical network layer, virtual network layer, and service layer of the target production environment in the mirror environment.

2) THE EVENTS REPRODUCTION DEPARTMENT

This department emulates the user's behavior and produces any needed faults in the reproduced environment.

3) THE RECOVERY DEPARTMENT

This department enables efficient recovery from artificially created faults.

4) THE DATA COLLECTION DEPARTMENT

This department collects all the data created in the mirror environment.

The details of AI training data collection using a mirror environment with four functional departments are as follows.

First, the target production environment is reconstructed in the environment reproduction department of the mirror environment, and the type and amount of training data needed depend on the target (AI model) [17].

Then, the events reproduction department is used to simulate the normal operation state of the target environment and the abnormal state of the target environment when a failure occurs by using the user access in the mirror environment in accordance with the type of AI training data required.

The data collection department is used to collect the data directly when the normal state data is needed. In the case of failure data, the fault recovery department is used to recover the failure after collection.

Finally, depending on the amount of AI training data required, the cycle of artificial fault occurrence and fault recovery is repeated until enough AI training data is collected.

C. ISSUE ANALYSIS

We have found five issues that need to be overcome to implement the proposed technology.

1) LOW-COST ENVIRONMENT REPRODUCTION

In the proposed method, the first step is reproducing the structure of the target production environment in the mirror

environment. Most network environments are very large. However, when the target production environment is huge, an identical composition is impractical to recreate because it is expensive. The increased cost of the environment can also significantly reduce the feasibility of the proposal. Thus, a low-cost environment needs to be made in accordance with the situation. Ways to reduce the mirror environment's building cost include using a device with lower performance than the target production environment or reducing the number of devices. For the mirror environment, we are mainly interested in collecting data for AI training. As long as the data to be collected (metrics, logs, and other data used for AI training) can be consistent, the same environment does not need to be replicated, only the necessary parts to reduce costs.

2) THE METHOD OF FAULT REPRODUCTION

When the AI needs fault data for training, the required faults need to be simulated and reproduced in the mirror environment. Initially, existing methods, such as the chaos engineering tool, can be used to simulate faults. However, the existing techniques can simulate only a very limited number of faults. Therefore, we need to explore the possibility of reproducing complex faults that occur less frequently. Furthermore, if complex faults (including unknown faults) cannot be simulated and reproduced by existing methods, methods other than existing methods are needed to reproduce and simulate the faults.

3) METHODS OF FAULT RECOVERY

In many cases, we need a large amount of training data for AI training. The data at the time of fault is often collected after an artificially created fault. Nevertheless, one fault is often not enough to produce data for training AI. To efficiently collect enough training data in a limited working time, multiple cycles of fault generation and fault recovery need to be performed efficiently. Therefore, an effective fault recovery method becomes essential. Considering the types of faults and the diversity and complexity of their recovery methods, we can save time by recording and backing up the state before the fault and not performing a one-to-one recovery after the fault but just rolling back to the state before the fault. However, the feasibility and details of this approach still need to be discussed.

4) MIGRATE AI

Once we have collected data in the mirror environment, we need to use the artificially generated data for AI training. However, we aim to solve the problem of insufficient AI training data in the target production environment, so the AI trained in the mirror environment needs to be migrated to the target production environment for use. There may be differences because the low-cost mirror environment and the target production environment are not the same. To enable the AI trained in the mirror environment to achieve the same effect as the AI trained directly in the target production environment, we think we also need the difference between the two environments to make adjustments to the AI so that the two environments are considered the same for the trained AI.

5) EVALUATION METHOD

A method is needed to accurately determine whether the AI trained in the mirror environment is the same as the AI trained in the target production environment. Collecting data is challenging in the real target production environment, so the mirror environment is needed to collect data for AI training. However, because of this, there is no way to compare the accuracy of the AI trained in the mirror environment with the AI trained in the target production environment. Although there are already some metrics (such as F_1-score) that can be used to evaluate AI, they cannot be used to evaluate the proposed technology. Therefore, an evaluation method for proposed technology also needs to be discussed.

IV. LOW-COST ENVIRONMENT REPRODUCTION

We thought that the first issue was to overcome the cost of setting up the mirror environment in Chapter III because the real production environment is often huge. Suppose we need to set up a mirror environment exactly like the target production environment. In that case, we may incur the same or even higher costs. In particular, if the target production environment is huge and the equipment is very aged, an identical environment (same device or same scale) is almost impossible to build. This problem may significantly reduce the feasibility of the proposed method. Thus, to improve the feasibility of the proposed method, we discuss the problem of building a mirror environment at a low cost in this chapter.

We made a hypothesis and designed a proof experiment to reduce the cost of setting up a mirror environment. We assumed that we only need to obtain the virtual layer data of the target production environment to train the AI that determines fault spots in its virtual layer. Therefore, even if the underlying physical network layer uses lower performance devices or fewer devices than the target production environment, as long as the virtual layer is identical, AI should perform the same (accuracy and other measures are similar).

To test this hypothesis, we built three virtual layers with identical environments using Raspberry Pi and a PC server with significant performance differences to set up two experimental control groups for comparison.

We designed Experiment I to compare AI training when the mirror environment is built with fewer devices than the target production environment but has the same virtual layers. We designed Experiment II to compare AI training when the mirror environment is built with lower performance devices than the target production environment but has the same virtual layer.

TABLE 1. Physical layer of experiment I.

Experiment I		
Environment	Target Production I	Mirror
Device	Raspberry Pi	
Number of devices	3	2
Operating system	Ubuntu 20.10	
CPU model	Broadcom BCM2711 Quad core Cortex-A72 (ARM v8) 64-bit@1.5GHz 8GB	
Cores	4	
Architecture	aarch64	

TABLE 2. Physical layer of experiment II.

Experiment II		
Environment	Target Production II	Mirror
Device	PC server	Raspberry Pi
Number of devices	2	
Operating system	Ubuntu 20.04.1 LTS	Ubuntu 20.10
CPU model	Intel(R)Xeon(R) CPU E3-1535M v6@3.10GHz Intel(R)Core(TM) i7-7700 CPU@3.60GHz	Broadcom BCM2711 Quad core Cortex-A72 (ARM v8) 64-bit@1.5GHz 8GB
Cores	8	4
Architecture	x86_64	aarch64

A. PHYSICAL LAYER EXPERIMENT ENVIRONMENT

1) EXPERIMENT I

First, we built a target production environment I with three Raspberry Pi devices and the mirror environment for the target production environment with two Raspberry Pi devices. Then, we collected data in the mirror environment to train AI models and evaluated the performance of the trained AI. Finally, the AI model trained in the mirror environment was tested in the mirror environment and target production environment I. Table 1 shows information of target production environment I and the mirror environment.

2) EXPERIMENT II

First, we built a target production environment II with two PC servers and the mirror environment of the target production environment with two Raspberry Pi devices (the mirror environment here is the same as in experiment I). Then, we collected data in the mirror environment to train AI models. Finally, the AI model trained in the mirror environment was tested in the mirror environment and target production environment II. Table 2 shows information of experimental target production environment II and the mirror environment.

B. VIRTUAL LAYER EXPERIMENT ENVIRONMENT

In experiments I and II, we built the same three-tier (or three-layer) architectural model on the Kubernetes platform (the environment reproduction department), which can automatically restart (the recovery department). The three-tier model for the experiments is indicated in Figure 3. It comprises three layers: the database layer uses MongoDB,

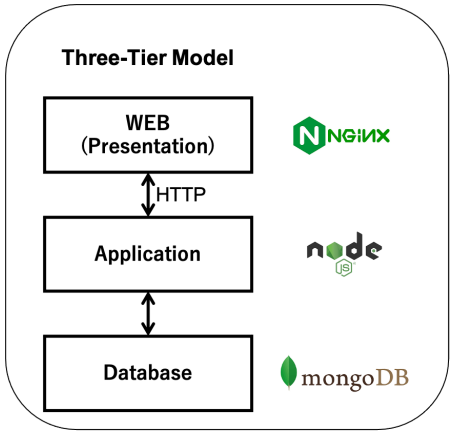


FIGURE 3. Three-tier architecture model.

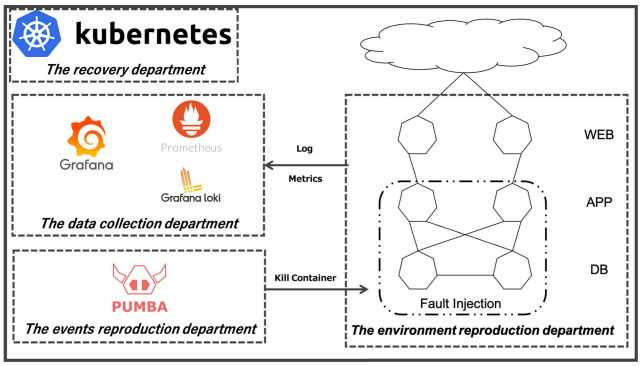


FIGURE 4. Configuration of the virtual layer.

the application layer uses Node.js to deploy an applet for logging in information about employees, and the web services layer (or presentation layer) uses NGINX to implement it.

In this experiment, six containers were used to build this three-tier model, and four (two for the app and two for the database (DB)) were chosen for fault injection. First, we used the chaos engineering tool called Pumba, an artificial fault generation tool (the events reproduction department), to randomly select one of the four containers to perform a forced stop (kill container) fault injection. Then we collected data from Pumba (fault times) and Prometheus (CPU and memory metrics data for 10s intervals) by Grafana (the data collection department) for a specific time interval, refer to Figure 4.

C. FAULT SPOT DETECTION AND RESULT MEASUREMENT

On the basis of the above environment composition, we collected data and trained AI in the mirror environment and evaluated the AI in target production environments I and II and the mirror environment. The experiments are demonstrated in Figure 5. We evaluated the AI by comparing data obtained in different environments. Four measurements were used for the evaluation [18], [19]:

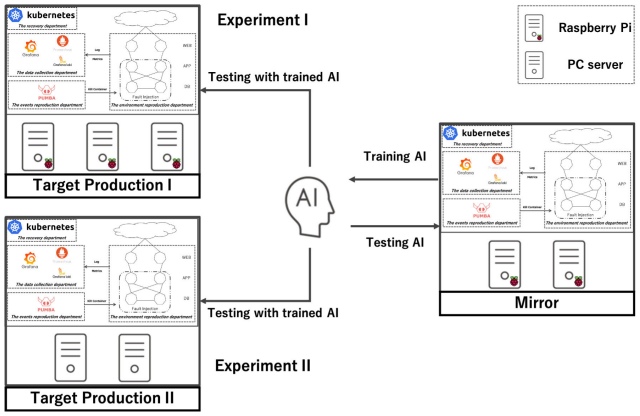


FIGURE 5. Image of experiments.

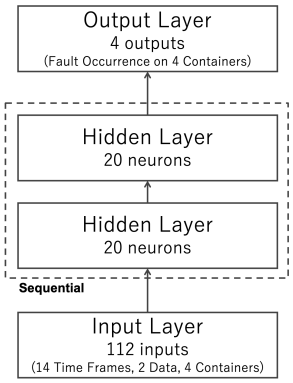


FIGURE 6. AI model of experiments.

1) Accuracy: percentage of correct results predicted by the trained AI out of the total data.

2) Precision: percentage of data in which the faults occurred where a trained AI predicted a [fault] that occurred.

3) Recall: percentage of data in which trained AI predicts [fault] out of the actual occurred fault data.

4) F_1-score: harmonic mean of precision and recall. Refer to (1).

$$F_1 = \frac{2 \times precision \times recall}{precision + recall}. \tag{1}$$

D. AI MODEL

The AI model used in the experiments is a NN using CPU and memory data as input data for fault spot detection. The composition of the AI model is demonstrated in Figure 6.

Input data is a vector of 112 dimensions. It is logs of the 7 moments before and after the specified moment (14 moments in total) and multiplied by 2 characteristics (CPU and memory) of 4 containers: 2*7*2*4 = 112.

From the chaos engineering tool Pumba logs, we obtain the time of the fault and use it as the specified moment.

The output is the probability of fault or non-fault of the 4 containers, expressed as a 4-dimensional vector. If the container's probability of failure (0 to 1) is higher than 0.5, the AI considers the container a faulted container.

The container determined to be faulted is marked as 1, and the ones determined not to be faulted are marked as 0. Moreover, because the faults injection method of this experiment only injects fault into one of the four containers each time, only one container is 1 at each moment, and the rest are 0. Thus, the labeling data is similar to (1 0 0 0).

TABLE 3. The evaluation results of experiment I&II.

Env	Measures	Containers				Average
		DB		APP		
		①	②	③	④	
I	Accuracy	1.000	1.000	0.986	0.991	0.994
	Precision	1.000	1.000	**0.875**	0.923	**0.995**
	Recall	1.000	1.000	1.000	1.000	1.000
	F_1-score	1.000	1.000	0.993	0.960	0.973
II	Accuracy	1.000	1.000	0.995	1.000	0.999
	Precision	1.000	1.000	0.996	1.000	0.992
	Recall	1.000	1.000	1.000	1.000	1.000
	F_1-score	1.000	1.000	0.982	1.000	0.996
Mirror	Accuracy	0.998	1.000	1.000	0.988	0.999
	Precision	1.000	1.000	1.000	0.988	0.997
	Recall	0.984	1.000	1.000	1.000	0.996
	F_1-score	0.992	1.000	1.000	0.994	0.997

E. RESULTS AND DISCUSSION

Finally, we used the AI model mentioned above for training and evaluation. We evaluated in the mirror environment experimental target production environments I and II, and the evaluation results are presented in Table 3.

From the results for experimental target production environments I and II, the four evaluation measures for each container are above 0.875, and the average values are even above 0.950. These results are similar to the evaluation results obtained in the mirror environment. This means that the AI trained in the mirror environment performs similarly to the others. This result shows that when an AI only needs the virtual layer's data for training, it is possible to train the AI with the same performance even if the physical layer is different, as long as the virtual layers that directly obtain the data are similar. However, the app used for the experiment is relatively simple and does not simulate user access, so further confirmation is needed, but this experiment proves this possibility.

Another point is that although the AI trained in the mirror environment is not poorly rated, this is not a criterion for judging how good the proposed method is. Even if the AI trained in the mirror environment does not have a high evaluation, the feasibility of our proposal can be proved by a similar evaluation in the production environment.

V. DISCUSSION

In this chapter, we will continue our comparative analysis of the low-cost mirror environment and use two simple experiments to discuss the feasibility of our proposed approach.

A. THE CONNECTION BETWEEN DATA VOLUME AND AI

We used the two environments and the AI model from Chapter IV to analyze the connection between AI and data.

The environment built by using two Raspberry PIs is named PI_2 (mirror environment), and the environment built with three Raspberry PIs is named PI_3 (production environment I).

The same amount of data (150 times to kill the containers) is collected in each of the two environments to train and test the AI model, and the results are shown in Figure 7.

In the training results of the two environments, the four evaluation metrics (accuracy, precision, recall, and F_1-score) of the AI gradually increase in accordance with the increase in the amount of AI training data.

The training results from the two environments show that when the amount of AI training data increases, the four-evaluation metrics of AI also increase gradually. This result shows that the increase in data helps to improve AI accuracy. Moreover, even the mirror environment built at low cost has such characteristics.

In the experiments, we also found that the test results (the four evaluation metrics) of AI started to change smoothly (Figure 7) around reaching 150 times in training.

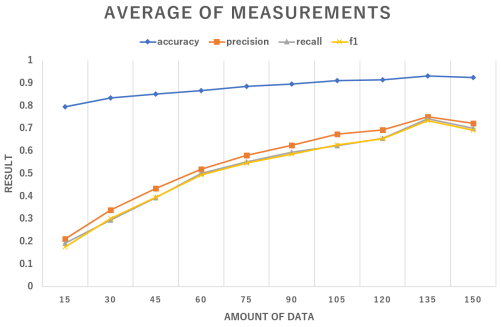
B. SIMULATE EXISTING METHODS AND COMPARE

To compare with existing techniques, in this section, we try to simulate existing techniques. We have chosen to simulate the reuse of data or results, the existing techniques in Chapter II-A-2), as an example to compare our proposal. This existing technique uses similar AI models trained in the past (pre-trained) and fine-tuned [20], [21] it with small amounts of target environment data to create the new AI model we need. However, as discussed in Chapter II, this approach assumes the existence of a suitable dataset or learned AI model that can be reused. A particularly well-matched dataset or model is often difficult to find. Thus, to simulate the reuse of the AI technique, we need to train a similar AI model from a mirror environment and fine-tune it with the production environment's (target environment) data.

First, on the basis of past data from our group business company, we set a data volume of 30 times as the data volume for the production environment to fine-tune the AI model. The results obtained in Section A show that the AI model in this experiment starts to stabilize when the training data reaches 150 times. Therefore, we decided to use 120 times data as training for a similar environment AI model and 30 times data in a production environment for fine-tuning. We collect data 120 times in the environment PI_2 as the similar environment to train the AI model and test the trained AI model in the environment PI_3 as the production environment. The results are shown in Table 4(before fine-tuning). This trained model is considered as a similar model that has been pre-trained. To simulate the existing model reuse method mentioned in Chapter II-A-2), we collected data in environment PI_3 30 more times to simulate the existing small amount of data. Then 30 times data were collected in the environment as a small amount of data from the existing production environment. Moreover, the trained past similar model models were fine-tuned to simulate the existing model re-training maneuvers. Finally, the fine-tuned AI model was tested in the PI_3 environment, and the results are shown in Table 4 (after fine-tuning).

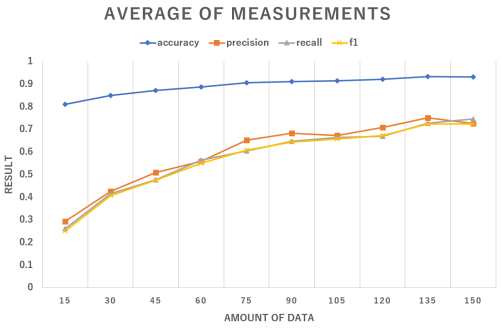
AI accuracy significantly increased when using the trained similar AI model and a small amount of data for fine-tuning, with the F_1-score increasing from 0.570 to 0.696. This result

Amount of data	Average			
	Accuracy	Precision	Recall	F_1-score
15	0.795	0.212	0.192	0.176
30	0.836	0.339	0.295	0.303
45	0.853	0.435	0.393	0.395
60	0.867	0.519	0.502	0.495
75	0.887	0.581	0.552	0.548
90	0.896	0.625	0.595	0.587
105	0.912	0.676	0.624	0.627
120	0.915	0.694	0.656	0.655
135	0.932	0.751	0.744	0.735
150	0.925	0.724	0.701	0.692



(a) Results of PI_2

Amount of data	Average			
	Accuracy	Precision	Recall	F_1-score
15	0.812	0.293	0.261	0.250
30	0.851	0.426	0.416	0.408
45	0.873	0.509	0.478	0.447
60	0.887	0.559	0.564	0.551
75	0.907	0.653	0.605	0.609
90	0.912	0.684	0.647	0.644
105	0.915	0.673	0.665	0.658
120	0.923	0.709	0.670	0.673
135	0.933	0.752	0.728	0.725
150	0.933	0.727	0.747	0.725



(b) Results of PI_3

FIGURE 7. Connection between AI and data.

TABLE 4. Simulation results of existing method.

Fine-tuning	Accuracy	Precision	Recall	F_1-score
Before	0.882	0.528	0.464	**0.570**
After	0.927	0.706	0.710	**0.696**

TABLE 5. Results of using mirror environment only.

Accuracy	Precision	Recall	F_1-score
0.993	0.718	0.807	**0.745**

TABLE 6. Results of discussion.

Sections		Data		AI Model	
		Collection Environment	Amount of data	Test Environment	F_1-Score
A		PI_3(Production Env)	150	PI_3(Production Env)	**0.725**
B	Before	PI_2(Similar Env)	120		0.570
(Fine-tuning)	After	PI_3(Production Env)	(120)+30		**0.696**
C		PI_2(Mirror Env)	350		**0.745**

means that with a small amount of data and a similar trained model, fine-tuning can result in a better AI model than training directly with a small amount of data.

Finally, we collected data from the mirror environment PI_2 350 times for AI model training and tested the AI in the production environment PI_3. It is effortless to obtain the data in the mirror environment. Therefore, we base our experiments on approximately two times the 150 times data in Sections A and Section B to set 350 times data for training the AI model. This is more than twice the amount of 150 times data. We think the mirror environment can easily obtain twice as much data or more, and more data has a better chance of training better AI models.

The results are presented in Table 5.

Comparing the results of the three experiments, shown in Table 6.

We found that the F_1-score of the AI model obtained when using only 150 times of data collected from environment PI_3 for training was 0.725. The F_1-score of the AI model obtained using the learned AI model and 30 times of data fine-tuning was 0.696. And the F_1-score of the AI model trained directly using the large amount of data (350 times) generated in the mirror environment PI_2 was 0.745. This result is higher than the combination of using a small amount of data and existing methods.

The results suggest that even when similar AI models are available for fine-tuning (which is often difficult to obtain in reality), generating more data directly in the mirror environment for AI training can yield better results if there is little available production environment data for fine-tuning.

VI. CONCLUSION

This paper proposes a mirror environment that creates the same data as the actual production environment. We think that collecting mirror environment-created data can compensate for network faults' insufficient AI training data.

To verify the feasibility of the proposed method, we conducted fault detection experiments with a virtual network layer, discussed the possibility of low cost using controlled experiments, and compared the discussed results with existing methods of fine-tuning. The comparison results prove that our proposed approach is a promising option to compensate for the insufficiency of AI training data for network operation and maintenance (O&M).

Following this, we plan to reduce the data discrepancy between a mirror environment and a real production environment by adding user access simulations. In addition, we plan to add more network failure types, increase the complexity of network failure modes, and even find new unknown network failure types. Try to use other AI outside of this paper to verify the feasibility of the proposed approach and the accuracy of the generated data. A usable mirror environment needs to be built. To build a usable mirror environment for collecting data, analyzing the relationship between the trained AI and the object production environment will also be one of our research focuses in the future.

REFERENCES

[1] K. Grace, J. Salvatier, A. Dafoe, and B. Zhang, "When will AI exceed human performance? Evidence from AI experts," *J. Artif. Intell. Res.*, vol. 62, pp. 729–754, Jul. 2018, doi: 10.1613/jair.1.11222.

[2] K. Johnson and A. Lawrence, "AI/ML in security orchestration, automation and response: Future research directions," *Intell. Automat. Soft Comput.*, vol. 28, pp. 534–537, 2021, doi: 10.32604/iasc.2021.016240.

[3] L. Lin, B. Liu, X. Zheng, and Y. Xiao, "An efficient image categorization method with insufficient training samples," *IEEE Trans. Cybern.*, early access, Aug. 11, 2020, doi: 10.1109/TCYB.2020.3011165.

[4] S. T. Fontanini, J. Wainer, V. Bernal, and S. Maragon, "Model based diagnosis in LANs," in *Proc. IEEE Workshop IP Oper. Manage.*, Oct. 2002, pp. 121–125, doi: 10.1109/IPOM.2002.1045767.

[5] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: A big data–AI integration perspective," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1328–1347, Apr. 2021, doi: 10.1109/TKDE.2019.2946162.

[6] J. Heidemann, K. Mills, and S. Kumar, "Expanding confidence in network simulations," *IEEE Netw.*, vol. 15, no. 5, pp. 58–63, Sep. 2001, doi: 10.1109/65.953234.

[7] A. Mikolajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *Proc. Int. Interdiscipl. PhD Workshop (IIPhDW)*, May 2018, pp. 117–122, doi: 10.1109/IIPHDW.2018.8388338.

[8] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, p. 60, Dec. 2019, doi: 10.1186/s40537-019-0197-0.

[9] F. Zhuang, Z. Qi, K. Duan, D. Xi, and Y. Zhu, "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021, doi: 10.1109/JPROC.2020.3004555.

[10] T. Hartmann, A. Moawad, C. Schockaert, F. Fouquet, and Y. Le Traon, "Meta-modelling meta-learning," in *Proc. ACM/IEEE 22nd Int. Conf. Model Driven Eng. Lang. Syst. (MODELS)*, Sep. 2019, pp. 300–305, doi: 10.1109/MODELS.2019.00014.

[11] J. Shao, C. Huang, Q. Yang, and G. Luo, "Reliable semi-supervised learning," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 1197–1202, doi: 10.1109/ICDM.2016.0159.

[12] A. Basiri, N. Behnam, R. de Rooij, L. Hochstein, L. Kosewski, J. Reynolds, and C. Rosenthal, "Chaos engineering," *IEEE Softw.*, vol. 33, no. 3, pp. 35–41, May 2016, doi: 10.1109/MS.2016.60.

[13] *ChaosMonkey*. Accessed: Nov. 15, 2021. [Online]. Available: https://github.com/Netflix/chaosmonkey

[14] *Pumba*. Accessed: Nov. 15, 2021. [Online]. Available: https://github.com/alexei-led/pumba

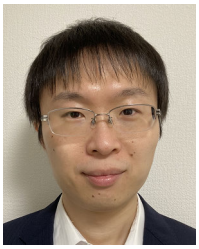
[15] *Gremlin*. Accessed: Nov. 15, 2021. [Online]. Available: https://www.gremlin.com

[16] C. Rosenthal, A. Blohowiak L. Hochstein, N. Jones, and A. Basiri, *Chaos Engineering–Building Confidence in System Behavior Through Experiments*. Newton, MA, USA: O'Reilly, 2017.

[17] I. V. Chugunkov, D. V. Kabak, V. N. Vyunnikov, and R. E. Aslanov, "Creation of datasets from open sources," in *Proc. IEEE Conf. Russian Young Researchers Electr. Electron. Eng. (EIConRus)*, Jan. 2018, pp. 295–297, doi: 10.1109/EIConRus.2018.8317091.
[18] B. J. M. Abma, "Evaluation of requirements management tools with support for traceability-based change impact analysis," Software Eng., Fac. Elect. Eng., Math. Comput. Sci., Univ. Twente, Sep. 2009.
[19] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1819–1837, Jun. 2014, doi: 10.1109/TKDE.2013.39.
[20] M. A. Wani and S. Afzal, "A new framework for fine tuning of deep networks," in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2017, pp. 359–363, doi: 10.1109/ICMLA.2017.0-135.
[21] T. Alshalali and D. Josyula, "Fine-tuning of pre-trained deep learning models with extreme learning machine," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Dec. 2018, pp. 469–473, doi: 10.1109/CSCI46756.2018.00096.

DI LI received the B.S. and M.S. degrees in information science and engineering from Ritsumeikan University, Shiga, Japan, in 2017 and 2019, respectively.

She joined NTT Access Network Service Systems Laboratories in 2019, where she is currently a a Research Engineer engaged in the research and development of access network operations and maintenance automation technology.



KAZUAK AKASHI received the B.S. and M.S. degrees from The University of Electro-Communications, in 2009 and 2011, respectively.

He joined NTT Access Network Service Systems Laboratories, where he is currently a Research Engineer engaged in the research and development of access network operations, and development, maintenance, and management of internal IT systems.

HARUHISA NOZUE received the M.S. degree in mathematical sciences from Nagoya University, Aichi, in 2003.

He joined NTT Access Network Service Systems Laboratories in 2003, where he is a Senior Research Engineer of the Access Network Operation Project, and is engaged in the research and development of operation support systems of access networks.

KENICHI TAYAMA received the B.S. and M.S. degrees from The University of Electro-Communications, in 1993 and 1995, respectively. He joined NTT Access Network Service Systems Laboratories, where he is a Group Leader, a Senior Research Engineer, and a Supervisor. Since then, he has been engaged in the research and development of access network operations, and planning and development of internal IT systems, and network operations and maintenance.