IEICE Communications Express, Vol.13, No.5, 159–162

ComEX

LETTER

Evaluation on eBPF-based network failure prediction using AutoGluon

Tianhao Zhu[1, a)], **Jiwon Lee**[1], **Bojian Du**[1], **Ryoma Kondo**[1], **Kentaro Matsuura**[1], **Hiroyuki Morikawa**[1], **and Yoshiaki Narusue**[1]

Abstract This study evaluates an extended Berkeley Packet Filter (eBPF)-based network failure prediction method using Autogluon-Tabular to process the fine-grained network information extracted by eBPF. The extracted information is considered as input features of the proposed model, which aims to predict the subsequent packet loss and determine a network failure event before it causes a huge impact. Supervised learning and semi-supervised learning are both adopted in Autogluon. The accuracy and detection time are evaluated as the main criteria. Simulation results show that F1 scores exceed 0.9 for our proposed method, and the proposed method can achieve prediction for potential failure events within 30 and 40 seconds when symptoms such as packet loss occur.

Keywords: network failure prediction, packet loss, eBPF, Autogluon, feature selection

Classification: Network

1. Introduction

In recent years, there has been a rapid development of network failure prediction methodologies as modern network frameworks become increasingly complex and uncertain. For network services like cloud-native functions (CNFs) and network function virtualization (NFV), network failures tend to become more complex and frequent [1], making proactive operations more essential. In reality, network failures such as gradual packet loss take a certain amount of time to impact service quality, which enables us to predict failures in advance to prevent potential service damage.

The extended berkeley packet filter (eBPF) is considered one of the top candidates to provide detailed data about the network for model construction. Observability is the most essential function provided by eBPF in terms of network failure prediction. Not only does eBPF enable the derivation of fine-grained network information from the Linux kernel space, but it is also capable of monitoring CNFs in 5G core network (5GC) [2]. By applying the eBPF, network data such as CPU queue latency and TCP retransmission count can be easily collected.

Many researchers have studied eBPF-based methods for addressing network failures. The authors of [3] proposed an eBPF-based segementation routing (SR) method to achieve flexible failure detection and rapid rerouting. Because the IPv6 data plane of SR (also called SRv6) faces technical issues in implementing NFV, eBPF enables it to execute virtualized functions directly inside the Linux kernel without modifying it. In [4], a network monitoring mechanism at the kernel level was introduced for the Kubernetes cluster in the Alibaba container service. The eBPF was used to provide observability at the kernel level and achieve better troubleshooting and fault location for a large number of applications. These studies explored the effectiveness of eBPFs in terms of network monitoring and observability. However, they did not fully utilize the collected network data and neglected their potential use in failure prediction. In fact, the study in [5] evaluated the failure prediction model of long short-term memory (LSTM) trained on fine-grained eBPF data in CNFs and demonstrated its effectiveness.

This study aims to provide a more thorough evaluation to confirm how early and accurate network failures in a containerized environment can be predicted by eBPF compared with previous works. Specifically, we consider multiple general machine learning (ML) models such as supervised and semi-supervised learning models, regression and classification models, instead of using only one ML model. We trained these models on the eBPF data by applying AutoGluon-Tabular, an open-source automated learning (AutoML) framework [6]. Both classification and regression were used for supervised learning. For semi-supervised learning, we used normal behavior modeling based on regression. Further, we conducted feature selection to increase the training efficiency, and successfully reduced the number of features to 200. The simulation results show significant progress with respect to the F1 score and prediction time.

2. eBPF based network feature information extraction

The eBPF was originally proposed to address packet filtering problems, and it has since been extended to other functions, such as network observability and security. This enables network programs to be conducted directly in the kernel space, saving time for transferring data to the user space, which better utilizes hardware resources and provides better efficiency for networking tasks such as packet filtering [2].

Network feature information can be extracted using eBPF. Because eBPF program is event-driven, eBPF code can be registered to a network packet kernel event and programmed to collect network data. In this case, when packet communication is established, fine-grained information such as CPU run queue latency, TCP life time and TCP retransmission can be successfully obtained. In addition, in terms of network information extraction, cAdvisor is also applicable. It is an open-source container monitoring tool provided by Google, combined with eBPF, and can capture network information in a layer lower than the containers.

[1] Dept. of Electrical Engineering and Information System, University of Tokyo, Bunkyo-ku, Tokyo 113-8656, Japan
[a)] zhuth34@hmlab.t.u-tokyo.ac.jp

DOI: 10.23919/comex.2023XBL0183
Received December 26, 2023
Accepted February 2, 2024
Publicized March 12, 2024
Copyedited May 1, 2024

However, not all data extracted by eBPFs are relevant to fault occurrences. The use of all data to build ML models can lead to an increased model size and longer learning and inference times, making it challenging to deploy them on edge servers with limited computing resources. Therefore, it is necessary to select effective feature data in order to shorten the learning time of models and even improve their accuracy by preventing high-dimensional input vector overfitting problems.

3. Experiment

3.1 Dataset

In this study, we utilized the dataset provided by KDDI in the ITU AL/ML in 5G Challenge 2022 [7]. The dataset was simulated in a CNF-based 5GC test environment. Figure 1 illustrates the behavior within one cycle of the simulation process. One cycle lasted for 700 seconds, and network features were collected every 10 seconds, resulting in 70 rows of datasets per cycle. Three types of feature were collected, namely cAdvisor, eBPF, and 5GC logs.

In one cycle, the first 100 seconds represents the preparation period during which the initialization of the 5GC network takes place. For normal cycles, which are cycles with only very small number of failures, UE attach and detach operations are repeated from 100 seconds to 700 seconds (600 seconds duration). The initial number of UEs is 200, and attach and detach operations occur between 200 and 220 UEs. For abnormal cycles, which are cycles with a certain number of network failures, packet loss is considered as the only failure type, starting at approximately 90-100 seconds and increasing linearly during the cycles. The entire dataset includes 75% of normal cycles and 25% of abnormal cycles. Failure prediction models are trained to raise alarms as early as possible for abnormal cycles, and do not cause false alarms throughout the entire cycle for normal ones.

3.2 Experiment procedure

Autogluon-Tabular was used to train the model, which can automatically select the submodels and adopt a multi-layer stack ensembling strategy to conduct training [6]. The selected submodels included neural networks, CatBoost and random forest. We compared the performance of the three most significant models in network failure prediction, which are regression model and classification model of supervised learning, and normal behavior model of semi-supervised learning. These three models can comprehensively reflect the feasibility of using the data extracted by the eBPF. By determining the best learning model, we systematically constructed an eBPF-based methodology. The F1 score and average detection time were utilized as the evaluation criteria in our models. The prediction time is defined as the earliest time at which the F1 score reaches 0.9, which indicates that the model has achieved a good performance in terms of both precision and recall.

In the regression model, the output is considered as the number of packet loss at the last row of each cycle. For input tunning, we conducted downsampling for all normal cycles and adopted the data at the early stages of failure only for abnormal cycles. In this process, a threshold determination is required. When the number of packet loss exceeds the threshold, the entire cycle is considered abnormal. Validation data with 100 cycles was created to determine the threshold, where the ratio of normal and abnormal cycles was maintained at 3:1. We changed the threshold from 20 to 60 and checked the F1 score at two different detection times: 130 seconds, which was the first time the F1 score exceeded 0.9, and 600 seconds, which corresponded to the last row of each cycle. Figure 2 shows that the regression model had F1 scores of 0.961 at 130 seconds in red line and 0.993 at 600 seconds in blue line when the threshold was 30. Therefore, we set the threshold to 30.

In the classification model, validation data for the threshold is no longer required because the final output of the models directly predicts the column “label” of each row, which shows whether the cycle is normal or abnormal. The data prepossessing method for the classification model was the same as that for the regression model.

For the semi-supervised learning, a normal behavior model that uses only rows of normal cycles to train the model was utilized [8]. The normal behavior model sets the model output of model as the number of packet loss, which is simliar to the regression model in supervised learning. However, the output was set as the number of failures at each row and the difference between the output of the trained model and the actual number of failures was used for evaluation. The threshold for this difference was set to be three times the standard deviation of the trained model according to the 3-sigma rule of the training data. Since the standard deviation was 1.118, the threshold was set to 3.35.

In addition, to select the most appropriate feature information, we use a feature importance ranking algorithm, which can reduce the number of input parameters and increase the

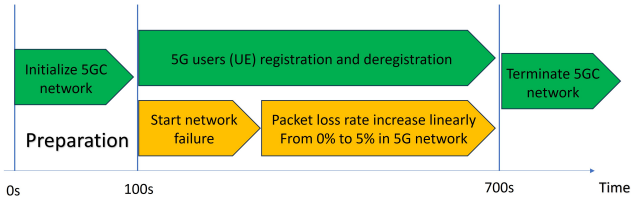

Fig. 1 Behavior within one cycle of simulation

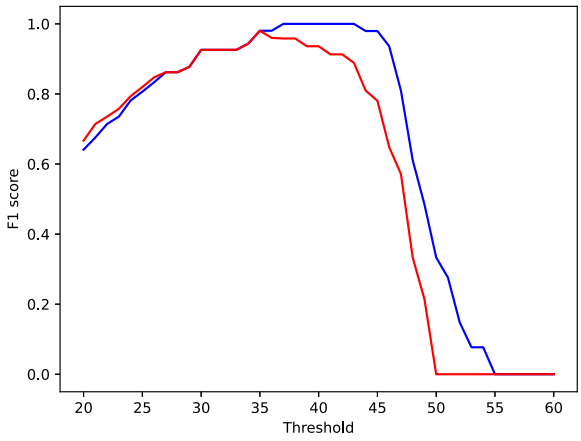

Fig. 2 Threshold determination for regression model

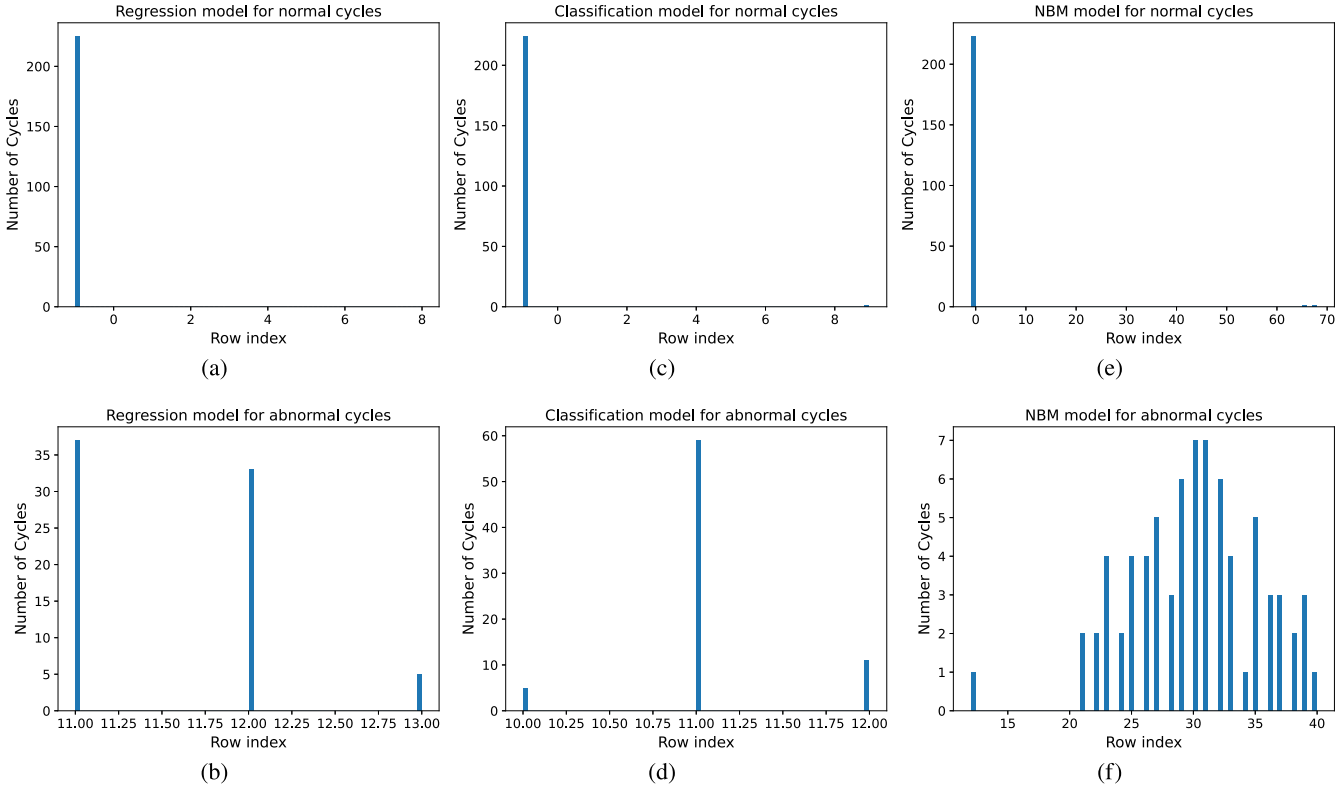

Fig. 3 (a)Normal cycles for regression model. (b)Abnormal cycles for regression model. (c)Normal cycles for classification model. (d)Abnormal cycles for classification model. (e)Normal cycles for normal behavior model. (f)Abnormal cycles for normal behavior model

calculation efficiency. By randomly selecting five normal cycles (5×60 rows = 300 rows) and two abnormal cycles (2×50 rows = 100 rows), while maintaining the 3:1 ratio, we compared the performance of the model with different numbers of features four time and ranked the feature importance. We then obtained a table showing the order of the feature importance.

3.3 Experimental result

3.3.1 Models trained with all metrics

The experimental results of the three proposed models using all the features are shown in Fig. 3. In these figures, the horizontal axis represents the row index for the time at which the cycle was determined to be abnormal, whereas the vertical axis represents the number of cycles. In addition, the row index is set as -1 when the cycle is considered as normal all the time. Figures 3(a) and (b) present the results of network failure prediction using the regression model. Approximately 98.7% of the normal cycles did not exceed the threshold, and retained the value of -1. In Fig. 3(b), it can be observed that the abnormal cycles first exceed the threshold between 10th-to the 12th rows, and approximatelhy 90.7% of abnormal cycles can be predicted within index 11, which is 120 seconds. The accuracy of the regression model was calculated and an F1 score of 0.922 was achieved at the prediction time of 120 seconds, and maintained a high F1 score (0.993) until the end of the cycle.

Figures 3(c) and (d) also show the good performance of the classification model. Figure 3(c) shows the results of the normal cycles, where approximately 99.6% of normal cycles were correctly classified. The results of the abnormal cycles are shown in Fig. 3(d), where abnormal cycles were predicted between rows 10 and 12, revealing that approximately 77.3% of the cycles could successfully predict the ongoing failure before index 12 (130 seconds).

For normal behavior model, the performance became relatively low, as shown in Figs. 3(e) and (f). In normal cycles, approximately 82.7% of them did not exceed the threshold of 3.35 throughout the entire cycle, and were correctly classified. However, the abnormal cycles exceed the threshold over a wide range of 0-45, which reveals that the prediction time of the network failure is performed later than supervised learning. Figure 3(f) shows that the F1 score was 0.905 at a time of 340 seconds, and 0.987 at 600 seconds. Because the normal behavior model does not use failure data, the failure prediction time is longer than that of supervised learning. As a result, the normal behavior model is only capable of determining a failure event for a cycle 240-250 seconds after the packet loss starts.

3.3.2 Models trained with selected features only

The results of feature importance selection are summarized in one table by calculating the average importance value and arranged in descending order. As shown in Table I, the types and number of features in the top 200 are presented in descending order. The three most important features are “tcpwin,” “tcprtt,” and “runqlat,” which are collected by the eBPF and reflect the state of TCP communication and the scheduling process of CPU.

We used the regression model of supervised learning as an example and employ the selected features to reconstruct it.

Table I Important features of the regression model

Important Feature	Number
eBPF tcpwin	44
eBPF tcprtt	40
eBPF runqlat	27
eBPF tcpretrans	15
cAdvisor interfaces	12
cAdvisor container memory rss	10
cAdvisor container memory working set bytes	10
cAdvisor container memory mapped file	9
cAdvisor container memory cache	6
cAdvisor container memory usage bytes	6
cAdvisor devices	4
cAdvisor container cpu usage seconds	3
5G SM PduSession	3
cAdvisor container cpu system seconds	2
cAdvsior container last seen	2
cAdvsior container cpu cfs periods	2
eBPF tcpdrop	1
eBPF pidnss	1
5G RM DeregSucc	1
cAdvisor container cpu user seconds	1
eBPF runqlen	1
Total	200

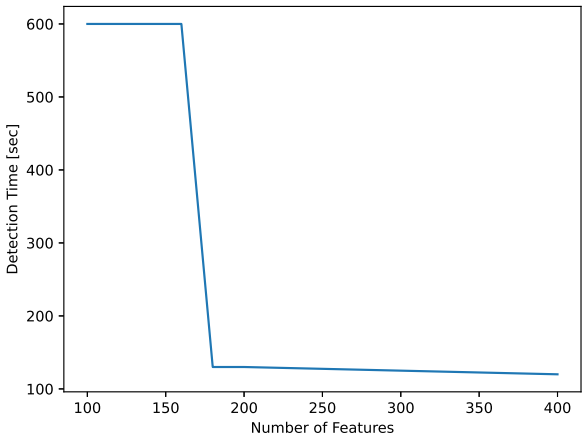

Fig. 4 Performance of models using selected features

Most of the parameter tuning was the same as that in the previous section; however only the selected features with high importance were adopted. When rebuilding the regression model, we compared the performance of different numbers of features among the top 400, 200, 180, 160, 150, 140, and 100 features. Figure 4 shows the relationship between the number of features used and the prediction time in the reconstructed regression model. In Fig. 4, the horizontal axis shows the number of features selected, and the vertical axis shows the prediction time at which the F1 score first exceeds 0.9. Specifically, the prediction time was 130 seconds when 400 or 200 variables were selected. When the number of features was reduced to 160 or 180, the time decreased slightly to 140 seconds. As a result, the use of the top 200 features is the most appropriate approach for realizing accurate prediction and a fast training speed simultaneously, which can be utilized in future work.

4. Conclusion

This study evaluated an eBPF-based network failure prediction method using Autogluon-Tabular and comprehensively determined its feasibility and performance with all general AI models. We compared the regression and classification models of supervised learning and normal behavior model of semi-supervised learning by adjusting the parameters in Autogluon. The results show that the network data extracted by the eBPF can realize fast and accurate network failure prediction when supervised learning is adopted. However, semi-supervised learning model still needs further improvement given its relatively low performance. Semi-supervised learning can work as an alternative when the amount of failure data is not enough. In addition, by evaluating feature importance, we prove that the top 200 network features selected by our algorithms can simultaneously achieve good performance and increase training efficiency.

Acknowledgments

This work was supported by KDDI Research, Inc.

References

[1] S. Imadali and A. Bousselmi, “Cloud native 5G virtual network functions: Design principles and use cases,” Proc. 2018 IEEE 8th International Symposium on Cloud and Service Computing (SC2), Paris, France, pp. 91–96, Nov. 2018. DOI: 10.1109/SC2.2018.00019

[2] S. Miano, M. Bertrone, F. Risso, M. Tumolo, and M.V. Bernal, “Creating complex network services with eBPF: Experience and lessons learned,” Proc. 2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR), Bucharest, Romania, pp. 1–8, June 2018. DOI: 10.1109/HPSR.2018.8850758

[3] M. Xhonneux and O. Bonaventure, “Flexible failure detection and fast reroute using eBPF and SRv6,” Proc. 2018 14th International Conference on Network and Service Management (CNSM), Rome, Italy, pp. 408–413, Nov. 2018.

[4] C. Liu, Z. Cai, B. Wang, Z. Tang, and J. Liu, “A protocol-independent container network observability analysis system based on eBPF,” Proc. 2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS), Hong Kong, pp. 697–702, Dec. 2020. DOI: 10.1109/ICPADS51040.2020.00099

[5] J. Kawasaki, D. Koyama, T. Miyasaka, and T. Otani, “Failure prediction in cloud native 5G Core with eBPF-based observability,” Proc. 2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring), Florence, Italy, pp. 1–6, June 2023. DOI: 10.1109/VTC2023-Spring57618.2023.10200028

[6] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, and A. Smola, “Autogluon-tabular: Robust and accurate automl for structured data,” arXiv preprint arXiv 2003.06505, 2020. DOI: 10.48550/arXiv.2003.06505

[7] ITU, “ML5G-PS-005: Network failure prediction on CNFs 5GC with Linux eBPF,” ITU, https://challenge.aiforgood.itu.int/match/matchitem/64/, Feb. 2022.

[8] A. Patcha and J.M. Liu, “An overview of anomaly detection techniques: Existing solutions and latest technological trends,” *Computer networks*, vol. 51, no. 12, pp. 3448–3470, 2007. DOI: 10.1016/j.comnet.2007.02.001