Machine Learning based Root Cause Analysis for SDN Network

Van TONG*, Sami SOUIHI*, Hai Anh TRAN† and Abdelhamid MELLOUK*
*Univ Paris Est Creteil, LISSI, F-94400 Vitry, France
Email: (van-van.tong, sami.souihi and mellouk)@u-pec.fr
† SOICT, HUST, Vietnam
Email: anhth@soict.hust.edu.vn

***Abstract*—Nowadays, the rapid growth of the Internet makes network management more complex due to various and complicated network problems. In the past, network administrators implemented troubleshooting approaches (e.g., ping, traceroute, etc.) manually to identify the root cause of problems. However, it is not effective due to human intervention and an increase of network devices. Consequently, the root cause analysis is considered by the research community. There are existing studies for the root cause analysis without human intervention (e.g., statistical approaches, heuristic algorithms, etc.). However, these approaches show limited performance (e.g., due to complex threshold identifcation, etc.). The emerging of machine learning (ML) and deep learning is a potential solution to overcome this obstacle, offering an opportunity to develop an effective root cause analysis approach. Therefore, in this paper, we propose a root cause analysis approach using ML and time-series network parameters to identify the root cause of problems in the network. In this approach, we consider balancing the accuracy and the time complexity of ML algorithms to select an appropriate ML technique. Moreover, we contribute troubleshooting datasets to identify three kinds of root causes including link failure, switch failure and buffer overload. The experimental results show that the proposal can achieve approximately 97 percent of precision, recall and f1-score in considered scenarios and require less processing time (only require 0.00143 ms for a sample) in comparison with other ML algorithms.**

***Index Terms*—Anomaly Detection, Root Cause Analysis, Machine Learning and SDN**

I. Introduction

Nowadays, the rapid development of the Internet results in more and more multimedia services (e.g., video streaming, online game, etc.) as well as diversified network devices such as IoT devices (Internet of Things). The huge number of network devices not only enlarges the network infrastructures but also leads to many critical issues in the network. According to Cerin et al. [1], a downtime on cloud-based services leads to negative economic impacts (e.g., the loss from over $30,000 to $6,700,000) for many service providers (e.g., Youtube, Facebook, Paypal, etc.). Therefore, network management and root cause analysis play an important role in the network.

In the past, administrators are able to troubleshoot network problems (e.g., using *ping*, *traceroute*, etc.) and solve it manually. However, it is not effective because of human intervention and the growth of network devices. Consequently, the root cause analysis is studied by the research community. Conventional root cause analysis (e.g., statistical approaches, heuristic algorithms, etc.) [2], [3] identifies the root cause using rules and policies with a specific threshold. However, an effective threshold identification is sometimes complicated. The emerging of machine learning and deep learning is a potential solution to solve this drawback. In fact, root cause analysis using machine learning (ML) and deep learning is studied by the research community to troubleshoot the network problems [4]–[6]. Benayas et al. [4] proposed a fault diagnosis method using Bayesian network and network parameters to troubleshoot the root cause of switch faults. Similarly, Hong et al. [5] proposed an anomaly detection method using machine learning to identify the root causes related to SLA (service-level agreement) violations for virtual network management. However, these studies have not considered a balance between the accuracy of machine learning algorithms and its processing time to select an appropriate ML algorithm.

In this paper, we propose a root cause analysis (RCA) mechanism using machine learning and time-series network parameters to detect an anomaly and its root causes (which network problems leading to the anomaly) in SDN environment. Nowadays, SDN is implemented by network operators (e.g., NTT [7], etc.) to facilitate the root cause analysis and network management. SDN with a separation between control plane and data plane offers opportunities to centralize its control as well as facilitate network monitoring with monitoring tools (e.g., link layer discovery protocol, etc.). The contributions of this paper are listed as follows:

- The balance between accuracy and processing time of different ML algorithms (e.g., random forest, gradient boosting, convolution neural network, etc.) is considered to select an appropriate algorithm for the root cause analysis. Unlike the existing studies that select the ML algorithm with the best performance, we select the machine learning algorithm with good performance and low time complexity to optimize the processing time when there is a huge amount of network traffic.
- Data collection related to abnormal situations is complex because it rarely happens in the network. Consequently, we implement a fault injection technique as proposed in [5], [6] to generate network problems and contribute

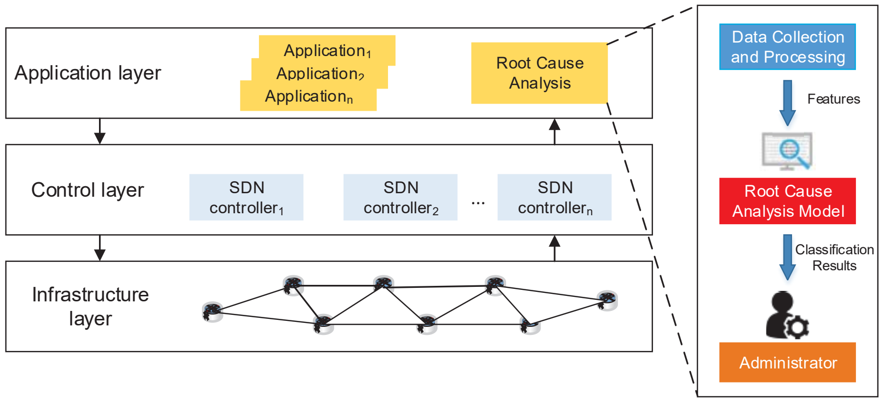

Fig. 1: Overall architecture of ML-based RCA in SDN environment.

troubleshooting datasets to identify three kinds of root causes including link failure, switch failure and buffer overload.

The remainder of the paper is organized as follows. Section II presents related work on anomaly detection and root cause analysis. The proposed ML-based RCA mechanism is discussed in section III. Section IV describes the experimental results of the proposal. The paper concludes with section V which highlights our future work.

II. Related Work

This section presents related work on anomaly detection and root cause analysis using statistical approaches and machine learning.

In the past, anomaly detection using statistical approaches is studied by the research community. Hochenbaum et al. [2] proposed an anomaly detection approach using statistical learning in cloud infrastructure. This approach calculates a test statistic value according to the median absolute deviation of time-series low-level system metrics and higher-level core drivers. If the test statistic value is bigger than a threshold, an anomaly happens in the network. Similarly, Chen et al. [3] proposed MADEL, an anomaly detection and localization approach using RTT (Round-trip Time) in NFV (Network Function Virtualization) environment. MADEL detects the anomaly by identifying the difference between anomalous and reference RTT matrix. After that, MADEL locates the paths that lead to the anomaly and narrows to sub-paths via monitoring a transmission time on these paths. However, identifying an adequate threshold for these approaches is sometimes complicated.

Consequently, many studies considered ML-based root cause analysis to detect the anomaly in the network. Benayas et al. [4] proposed a fault diagnosis mechanism using a Bayesian network in SDN. The objective is to infer ten kinds of problems (e.g., changing flow priorities, modifying import rules, etc.) thanks to Bayesian network and network parameters (e.g., change in a number of hosts, change in time-out, etc.). Kawasaki et al. [8] proposed a ML-based fault classification to analyze the root cause of failures in the NFV environment. This approach collects 41 features from the network environment and analyzes these features using ML algorithms (e.g., random forest, support vector machine, etc.) to identify three kinds of root causes including node-down, interface-down and CPU overload. Similarly, Hong et al. [5] proposed an anomaly detection method using machine learning to identify the root causes related to resource usage and SLA violations. This method identifies the root causes (e.g., high CPU utilization, lack of memory, etc.) based on 25 features and ML algorithms including distributed random forest, gradient boosting, extreme gradient boost and deep learning. However, these studies have not considered the balance between the accuracy and the time complexity of ML algorithm. Therefore, in this paper, we consider this aspect to select the appropriate ML algorithms. Besides, we contribute the troubleshooting datasets to identify the root causes including link failure, switch failure and buffer overload.

III. Proposal: ML-based RCA Mechanism

The overall architecture of ML-based RCA in the SDN environment is depicted as in Fig. 1. First, network traffic is collected from the infrastructure layer and processed in Data Collection and Processing module to extract network features. Then, these features are analyzed in Root Cause Analysis module to identify the root causes of problems. Finally, the Root Cause Analysis module notifies alarms to the administrators to operate the network effectively.

1) Data Collection and Processing: This module aims to collect and extract network features corresponding to the root causes for the RCA module. According to a report of network operators [9], we consider three root causes including link failure, switch failure and buffer overload. It is not easy to get datasets related to these root causes because it happens unpredictably in the network. Therefore, many studies [5], [6] use a fault injection technique to generate root causes. Link failure is simulated by adding latency in links while switch failure is simulated by generating rule failures which lead to packet loss in the switches. Buffer overload is simulated by sending a huge amount of traffic exceeding link's capacity. These root causes are simulated in two scenarios including static and dynamic networks. In static network, the delay and loss in the links are fixed with a specific value. In dynamic network, the link state is changed between normal and error state according to Gilbert-Elliot (GE) model [10] following a probability value. In this model, the probability of changing from normal to error state is p while the figure for changing from error to normal state is r. When a link is in the error state, the loss and delay will be modified as described in Tab. I.

TABLE I: Considered network conditions.

Root Causes	Static Network	Dynamic Network
Link failure	Delay: 125ms	Delay: [25, 50, 75, 100, 125ms]
Switch failure	Loss: 50%	Loss: [10, 20, 30, 40, 50%]
Buffer overload	Bandwidth: 10 mbps, Sending rate: 60-100 mbps	Bandwidth: 10 mbps, Sending rate: 60-100 mbps

In each network condition, we first extract nine features corresponding to network and standard parameters thanks to existing studies [11], [12] and API *PortStatistics* [13] in the controller. These parameters include latency, packet loss, link utilization, number of packet sent, number of packet received, number of byte sent, number of byte received, number of flow entries in switch and QoE (Quality of Experience). Then, these parameters on each link will be aggregated to calculate the features for the paths in order to identify the root cause of problems. After that, these features are normalized into a value between 0 and 1 according to a max normalization [14]. In RCA, we consider time-series features, and a sample aggregates features of ten consecutive time steps to identify the root causes.

TABLE II: The number of samples for each root cause in two troubleshooting datasets.

Root Causes	Static Network	Dynamic Network
Buffer overload	6900	6900
Link failure	9995	19710
Switch failure	7560	17565

We built two datasets for network troubleshooting corresponding to two network conditions including static and dynamic networks. The number of samples for each root cause in these datasets is depicted as in Tab. II. Each sample contains nine features of ten consecutive time steps.

2) ML-based RCA Method: The collected features from the Data Collection and Processing module are analyzed in a ML-based RCA method to identify the root cause of problems. The existence of problems leads to a fluctuation of these features. According to ML algorithms, this fluctuation can be detected to identify its root cause. The detail of this approach is described as in Fig. 2.

There are two main phases in this method including training and testing. In the training phase, the network features will be analyzed according to a ML algorithm to obtain a pre-trained RCA model for the testing phase. The objective of RCA model aims to infer the root causes in the network. There are different kinds of ML algorithms for the RCA. Therefore, we evaluate several algorithms to select the appropriate one and balance between the accuracy and the time complexity. The considered ML algorithms includes Support Vector Machine (SVM) [15], Bagging [16], Random Forest (RF) [15], Adaboost [17], Gradient Boosting [18] and Convolutional Neural Network (CNN) [19]. Regarding the CNN model, we use two convolutional 1D (1 dimensional) layer, a max pooling 1D, a flatter layer and three fully connected layers. For the other algorithms, we adjust hyper-parameters such as number of estimators, depth of tree and so on. These configurations are chosen as in existing studies [5] to optimize the performance of ML algorithms.

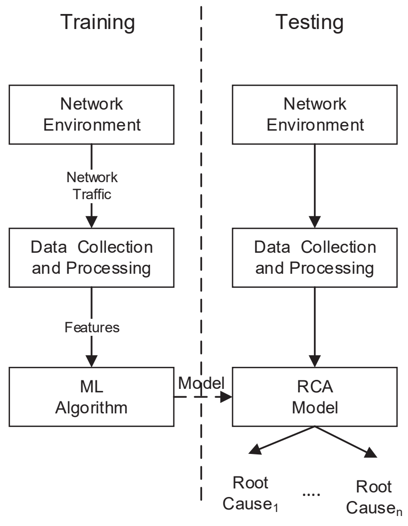

Fig. 2: The ML-based RCA Method.

IV. Experimental Results

A. Experimental Setup

In the experiments, we use emulation scenarios with tool *mininet* [20] to emulate network topology which contains five spine nodes and two leaf nodes. The delay and loss on each link are configured thanks to *mininet*. In GE model, p and r are set to 0.81 and 0.07 as in [10]. The ML algorithms are implemented according to a library *scikit-learn* [21] in Python. The troubleshooting datasets are divided into three parts. Training, validation and testing phases comprise 76, 4 and 20 percent of the datasets, respectively. The datasets and source code of the proposal are published in [22].

Performance of ML algorithms is evaluated thanks to performance metrics including precision, recall, and f1-score [23]. Precision is the percentage of relevant flows that are retrieved, while recall is the percentage of retrieved flows that are relevant. F1-score (F-measure) represents a harmonic mean between precision and recall. The detail of these parameters are depicted in Equ. 1, 2, 3. There are two ways to evaluate the quality of the overall classification including micro-averaging and macro-averaging value. In macro-averaging, a metric is averaged over all classes that are treated

TABLE III: Performance metrics of the considered ML algorithms for the dataset in static network.

Class	Precision						Nb Samples
	SVM	Bagging	RF	Adaboost	CNN	Gradient Boosting	
Buffer overload	0.8040	0.8525	0.9380	0.9637	0.9593	0.9675	1380
Link failure	0.9007	0.9802	0.9715	0.9670	0.9658	0.9764	1999
Switch failure	0.9739	0.9875	0.9776	0.9617	0.9834	0.9791	1512
Micro	**0.8920**	0.9403	**0.9634**	0.9644	**0.9691**	**0.9746**	4891
Macro	**0.8928**	0.9401	**0.9624**	0.9641	**0.9695**	**0.9743**	4891

Class	Recall						Nb Samples
	SVM	Bagging	RF	Adaboost	CNN	Gradient Boosting	
Buffer overload	0.8797	0.9928	0.9870	0.9819	0.9899	0.9928	1380
Link failure	0.8844	0.9415	0.9560	0.9540	0.9745	0.9735	1999
Switch failure	0.9134	0.8909	0.9517	0.9623	0.9431	0.9597	1512
Micro	**0.8920**	0.9403	**0.9634**	0.9644	**0.9691**	**0.9746**	4891
Macro	**0.8925**	0.9417	**0.9649**	0.9661	**0.9692**	**0.9753**	4891

Class	F1-score						Nb Samples
	SVM	Bagging	RF	Adaboost	CNN	Gradient Boosting	
Buffer overload	0.8401	0.9173	0.9619	0.9727	0.9743	0.9928	1380
Link failure	0.8925	0.9604	0.9637	0.9605	0.9701	0.9735	1999
Switch failure	0.9427	0.9367	0.9645	0.9620	0.9629	0.9597	1512
Micro	**0.8920**	0.9403	**0.9634**	0.9644	**0.9691**	**0.9746**	4891
Macro	**0.8918**	0.9382	**0.9633**	0.9651	**0.9691**	**0.9753**	4891

equally whereas micro-averaging is based on the cumulative True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) of the dataset.

$$Precision = \frac{TP}{TP + FP}. \quad (1)$$

$$Recall = \frac{TP}{TP + FN}. \quad (2)$$

$$F1 - score = \frac{2}{1/Recall + 1/Precision}. \quad (3)$$

B. Performance Analysis

1) Dataset in Static Network: There are two datasets including datasets in static and dynamic networks. First, we evaluate the balance between the accuracy and the time complexity of ML algorithms to select the appropriate one with the dataset in static network. The performance metrics of these algorithms are described in Tab. III. The ensemble learning is a model that makes predictions according to different models. There are two popular ensemble methods including bagging and boosting. The bagging ensemble method trains individual models on a different subset of datasets in parallel and aggregate to improve the performance. In contrast, the boosting ensemble method trains individual models in a sequence, and each model learns mistakes from the previous one to enhance the performance. In this paper, Bagging algorithm uses SVM as a base classifier, so its F1-score is higher than the figure for SVM. RF is the ML algorithm using the bagging ensemble method and decision tree as the individual model, so its F1-score (approximately 96.3 percent) is higher than the figure for SVM and Bagging.

TABLE IV: Time complexity of ML algorithms in RCA for the dataset in static network.

Algorithms	Training Time (ms)	Testing Time (ms)
SVM	**1.21472**	**0.49883**
Bagging	37.9581	35.8988
RF	**0.03084**	**0.00143**
Adaboost	0.36427	0.01378
CNN	**26.044**	**0.02636**
Gradient Boosting	**1.48182**	**0.00555**

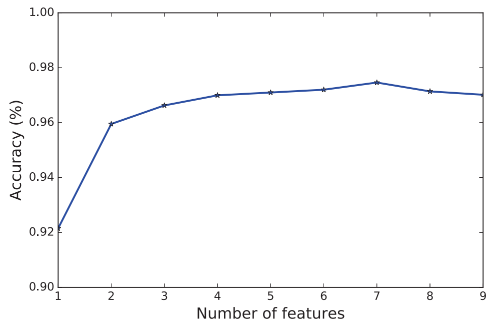

Fig. 3: The accuracy against the number of features in the feature selection method.

Adaboost and Gradient Boosting are the ML algorithms using boosting ensemble method and decision tree as the individual model. The objective of boosting ensemble method is to learn from the mistakes. Adaboost learns from the mistakes by increasing weight of misclassified samples while Gradient Boosting uses the gradient instead of adjusting the weight.

TABLE V: Performance metrics of the considered ML algorithms for the dataset in dynamic network.

Class	Precision						Nb Samples
	SVM	Bagging	RF	Adaboost	CNN	Gradient Boosting	
Buffer overload	0.6710	0.7813	0.9191	0.9201	0.9040	0.9535	1380
Link failure	0.7797	0.8669	0.9471	0.9244	0.9483	0.9487	3942
Switch failure	0.8492	0.8554	0.9451	0.9326	0.9292	0.9578	3513
Micro	**0.7914**	0.8491	**0.9418**	0.9269	**0.9333**	**0.9530**	8835
Macro	**0.7666**	0.8345	**0.9371**	0.9257	**0.9272**	**0.9534**	8835

Class	Recall						Nb Samples
	SVM	Bagging	RF	Adaboost	CNN	Gradient Boosting	
Buffer overload	0.5659	0.7739	0.9464	0.9341	0.9688	0.9667	1380
Link failure	0.8630	0.8757	0.9500	0.9434	0.9302	0.9619	3942
Switch failure	0.7996	0.8488	0.9308	0.9055	0.9229	0.9377	3513
Micro	**0.7914**	0.8491	**0.9418**	0.9269	**0.9333**	**0.9530**	8835
Macro	**0.7428**	0.8328	**0.9424**	0.9277	**0.9406**	**0.9554**	8835

Class	F1-score						Nb Samples
	SVM	Bagging	RF	Adaboost	CNN	Gradient Boosting	
Buffer overload	0.6140	0.7776	0.9325	0.9270	0.9353	0.9601	1380
Link failure	0.8192	0.8712	0.9486	0.9338	0.9392	0.9553	3942
Switch failure	0.8236	0.8521	0.9379	0.9188	0.926	0.9476	3513
Micro	**0.7914**	0.8491	**0.9418**	0.9269	**0.9333**	**0.9530**	8835
Macro	**0.7523**	0.8336	**0.9397**	0.9266	**0.9335**	**0.9543**	8835

Hence, Gradient Boosting is more flexible than Adaboost. As a result, the micro and macro F1-score of Gradient Boosting is slightly higher than the figure for Adaboost. Moreover, the F1-score of CNN is nearly similar to Gradient Boosting with 97 percent.

Despite the good performance, Adaboost, CNN and Gradient Boosting require much processing time. The time complexity of the considered ML algorithms is depicted as in Tab. IV. This table describes the average processing time for a sample in the training and testing phase. The training and testing time of RF are the lowest in the considered ML algorithms with 0.03084 and 0.00143 ms, respectively. The testing time of Adaboost, CNN and Gradient Boosting are nearly 10, 18 and 4 times the testing time of RF while the difference in F1-score between these algorithms is less than 1 percent. Therefore, we consider RF as a ML algorithm for RCA.

We use nine features to identify the root cause of problems, but some of these features are ineffective for the RCA. Consequently, we implement a feature selection method (wrapper method) to identify an appropriate feature set. The wrapper method [24] evaluates all possible feature sets based on a specific machine learning algorithm and selects the feature set with the highest accuracy. RF is considered as a learning algorithm in the wrapper method due to the balance between its accuracy and time complexity. Fig. 3 illustrates the accuracy against the number of features in the feature selection method. The feature set with seven features indicates better results than the others, so this feature set is considered in the RCA. This feature set contains latency, link utilization, number of packet received, number of byte sent, number of byte received, number of flow entries in switch and QoE. Tab. VI shows the F1-score of the selected feature set (Feature Set 2) in comparison with the feature set with all nine features (Feature Set 1).

TABLE VI: F1-score of two feature sets in the RCA for the dataset in static network.

Class	F1-score		Nb Samples
	Feature Set 1	Feature Set 2	
Buffer overload	0.9619	0.9748	1380
Link failure	0.9637	0.9710	1999
Switch failure	0.9645	0.9674	1512
Micro	**0.9634**	**0.9710**	4891
Macro	**0.9633**	**0.9710**	4891

TABLE VII: Time complexity of ML algorithms in RCA for the dataset in dynamic network.

Algorithms	Training Time (ms)	Testing Time (ms)
SVM	**3.70088**	**1.43646**
Bagging	81.4241	67.4686
RF	**0.38460**	**0.00587**
Adaboost	0.35830	0.01368
CNN	**26.1040**	**0.02455**
Gradient Boosting	**1.40897**	**0.01087**

2) Dataset in Dynamic Network: Tab. V and VII illustrate the performance metrics and the time complexity of ML algorithms with the selected feature set in the RCA for dynamic network. Similar to the dataset in static network, micro and macro f1-score of Gradient Boosting are the highest (approximately 95 percent) while the testing time of RF is the lowest in the considered ML algorithms for the dataset in dynamic network. The difference of f1-score between RF and Gradient Boosting is approximately 1 percent, so RF is considered as a ML algorithm in RCA for the dataset in dynamic network. The RCA can achieve good results in dynamic network with over 93 percent of the micro and macro precision, recall and f1-score. The micro and macro f1-score in dynamic network are approximately 94 percent, and reduce by 3 percent compared to the figure for static network. The reason is that the error status as well as the delay and loss value change after a given time in dynamic network. This leads to less fluctuation in the network features, and therefore reduces these performance metrics.

V. Conclusion

This paper proposed a root cause analysis approach using machine learning and time-series network parameters to identify the root cause of problems in the SDN environment. There are various ML algorithms, so we consider the balance between the accuracy and the time complexity to select the appropriate algorithm. Besides, we contribute the troubleshooting datasets to detect three root causes: link failure, switch failure and buffer overload. The experimental results illustrate that the precision, recall and f1-score of the proposal are approximately 97 percent in static network.

In the future, we will extend the troubleshooting datasets to take into account more root causes. Besides, there is a massive amount of valuable resources (e.g., system log, etc.) in addition to the network parameters to enhance the performance of RCA. Therefore, system logs as well as data-processing applications (e.g., Hadoop, Splunk, etc.), will be considered to improve the performance of RCA.

Acknowledgment

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.02-2019.314.

References

[1] C. Cérin, C. Coti, P. Delort, F. Diaz, M. Gagnaire, Q. Gaumer, N. Guillaume, J. Lous, S. Lubiarz, J. Raffaelli *et al.*, "Downtime statistics of current cloud solutions," *International Working Group on Cloud Computing Resiliency, Tech. Rep*, vol. 1, p. 2, 2013.

[2] J. Hochenbaum, O. S. Vallis, and A. Kejariwal, "Automatic anomaly detection in the cloud via statistical learning," *arXiv preprint arXiv:1704.07706*, 2017.

[3] J. Chen, M. Chen, X. Wei, and B. Chen, "Matrix differential decomposition-based anomaly detection and localization in nfv networks," *IEEE Access*, vol. 7, pp. 29 320–29 331, 2019.

[4] F. Benayas, A. Carrera, and C. A. Iglesias, "Towards an autonomic bayesian fault diagnosis service for sdn environments based on a big data infrastructure," in *2018 Fifth International Conference on Software Defined Systems (SDS)*. IEEE, 2018, pp. 7–13.

[5] J. Hong, S. Park, J.-H. Yoo, and J. W.-K. Hong, "Machine learning based sla-aware vnf anomaly detection for virtual network management," in *2020 16th International Conference on Network and Service Management (CNSM)*. IEEE, 2020, pp. 1–7.

[6] J. A. Cid-Fuentes, C. Szabo, and K. Falkner, "Adaptive performance anomaly detection in distributed systems using online svms," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 5, pp. 928–941, 2018.

[7] NTT, "Ntt communications extends sd-wan to more than 190 countries," May 2021. [Online]. Available: https://www.lightwaveonline.com/network-automation/article/16673711/ntt-communications-extends-sdwan-to-more-than-190-countries

[8] J. Kawasaki, G. Mouri, and Y. Suzuki, "Comparative analysis of network fault classification using machine learning," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–6.

[9] H. Zeng, P. Kazemian, G. Varghese, and N. McKeown, "A survey on network troubleshooting," *Technical Report Stanford/TR12-HPNG-061012, Stanford University, Tech. Rep.*, 2012.

[10] A. Bildea, O. Alphand, F. Rousseau, and A. Duda, "Link quality estimation with the gilbert-elliot model for wireless sensor networks," in *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, 2015, pp. 2049–2054.

[11] Y. Li, Z.-P. Cai, and H. Xu, "Llmp: exploiting lldp for latency measurement in software-defined data center networks," *Journal of Computer Science and Technology*, vol. 33, no. 2, pp. 277–285, 2018.

[12] L. Amour, V. Tong, S. Souihi, H. A. Tran, and A. Mellouk, "Quality estimation framework for encrypted traffic (q2et)," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[13] ONOS, "Portstatistics api," April 2021. [Online]. Available: http://api.onosproject.org/1.5.1/org/onosproject/net/device/PortStatistics.html

[14] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Applied Soft Computing*, vol. 97, p. 105524, 2020.

[15] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Rahim, "Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection," *IEEE access*, vol. 6, pp. 33 789–33 795, 2018.

[16] A. Niranjan, A. Prakash, N. Veena, M. Geetha, P. D. Shenoy, and K. Venugopal, "Ebjrv: An ensemble of bagging, j48 and random committee by voting for efficient classification of intrusions," in *2017 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*. IEEE, 2017, pp. 51–54.

[17] A. Rehman Javed, Z. Jalil, S. Atif Moqurrab, S. Abbas, and X. Liu, "Ensemble adaboost classifier for accurate and fast detection of botnet attacks in connected vehicles," *Transactions on Emerging Telecommunications Technologies*, p. e4088, 2020.

[18] J. Frery, A. Habrard, M. Sebban, O. Caelen, and L. He-Guelton, "Efficient top rank optimization with gradient boosting for supervised anomaly detection," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2017, pp. 20–35.

[19] D. Kwon, K. Natarajan, S. C. Suh, H. Kim, and J. Kim, "An empirical study on network anomaly detection using convolutional neural networks," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 1595–1598.

[20] D. Dholakiya, T. Kshirsagar, and A. Nayak, "Survey of mininet challenges, opportunities, and application in software-defined network (sdn)," in *International Conference on Information and Communication Technology for Intelligent Systems*. Springer, 2020, pp. 213–221.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[22] V. Tong, May 2021. [Online]. Available: https://github.com/vanvantong/rca-ml

[23] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information processing & management*, vol. 45, no. 4, pp. 427–437, 2009.

[24] Q. Al-Tashi, H. M. Rais, S. J. Abdulkadir, S. Mirjalili, and H. Alhussian, "A review of grey wolf optimizer-based feature selection methods for classification," *Evolutionary Machine Learning Techniques*, pp. 273–286, 2020.