Towards Building Intrusion Detection Systems for Multivariate Time-Series Data

ChangMin Seong[1], YoungRok Song[2], Jiwung Hyun[2], Yun-Gyung Cheong[2]

[1]Department of Computer Software, Sungkyunkwan University, South Korea

[2]Department of Artificial Intelligence, Sungkyunkwan University, South Korea

Abstract. Recent network intrusion detection systems have employed machine learning and deep learning algorithms to defend against dynamically evolving network attacks. While most previous studies have focused on detecting attacks which can be determined based on a single time instant, few studies have paid attention to subsequence outliers, which require inspecting consecutive points in time for detection. To address this issue, this paper applies a time-series anomaly detection method in an unsupervised learning manner. To this end, we converted the UNSW-NB15 dataset into the time-series data. We carried out a preliminary evaluation to test the performance of the anomaly detection on the created time-series network dataset as well as on a time-series dataset obtained from sensors. We analyze and discuss the results.

Keywords: Time Series, Intrusion Detection System, Stacked RNN, Unsupervised Learning, Anomaly Detection

1 Introduction

Due to the rapid development and popularization of networks, security issues are also becoming an important issue. In order to solve these security issues, a network intrusion detection system (NIDS) has been widely used. A NIDS is a system that reads network packets and detects attack traffic and is known as an effective defense method against network security issues. During the last decade, network security systems have been developed by employing various time-series intrusion detection techniques. Pankaj et.al [21] propose a Long Short Term Memory Networks based Encoder-Decoder scheme for Anomaly Detection (EncDec-AD) that learns to reconstruct normal time-series behavior. Kyle et. al [22] demonstrate the effectiveness of LSTM and propose dynamic thresholding approach using LSTMs. Ding et. al [23] propose a real-time anomaly detection algorithm (RADM) based on Hierarchical Temporal Memory (HTM) and Bayesian Network (BN). Park et. al [24] introduced a long short-term memory-based variational autoencoder (LSTM-VAE) that fuses signals and reconstructs expected distribution.

Furthermore, unsupervised learning algorithms have been getting more attention owing to their advantage of training the models without labels during the training phase [11, 12]. In the unsupervised methods, attacks are generally detected by regarding them as outliers or anomalies. More details about outlier detection can be found in [1, 2, 10].

Time-series data mean the data annotated with time stamps, collected at regular time intervals. Depending on what is considered an outlier, time-series outliers are largely divided into two types: point outliers and subsequence outliers [2]. A point outlier means an outlier of which value is significantly different from the values of the surrounding data in the overall flow of data in time order as shown in Figure 1. In the figure, a point between 10 and 11 can be regarded as normal with a global perspective where similar data values exist between 21 and 22, but it is determined as an outlier considering the values of its neighbors with a local perspective [3]. These outliers can be determined relying on their characteristics at a specific time instant.

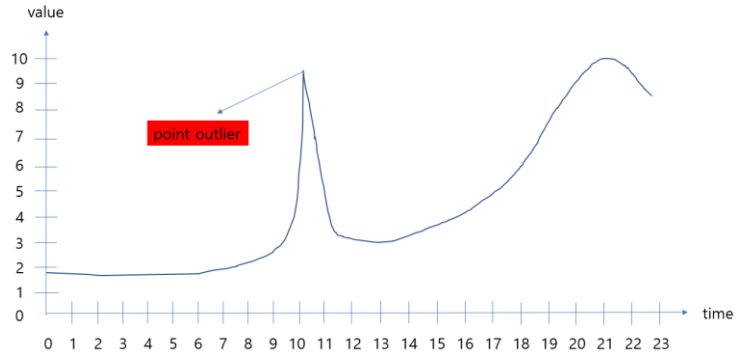

Fig. 1. An illustration of a point outlier where samples between 10 and 11 are spiking, distinguished from their neighboring data.

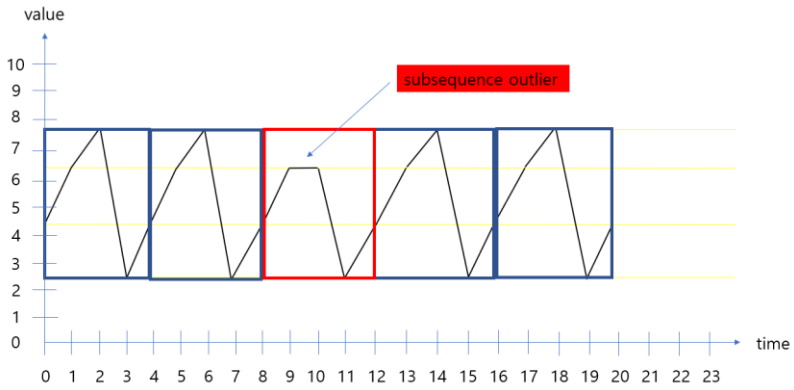

Fig. 2. An illustration of a subsequence outlier which is represented in the red box. The data values are within the minimum and the maximum of normal data, and yet the overall pattern is different from the rest.

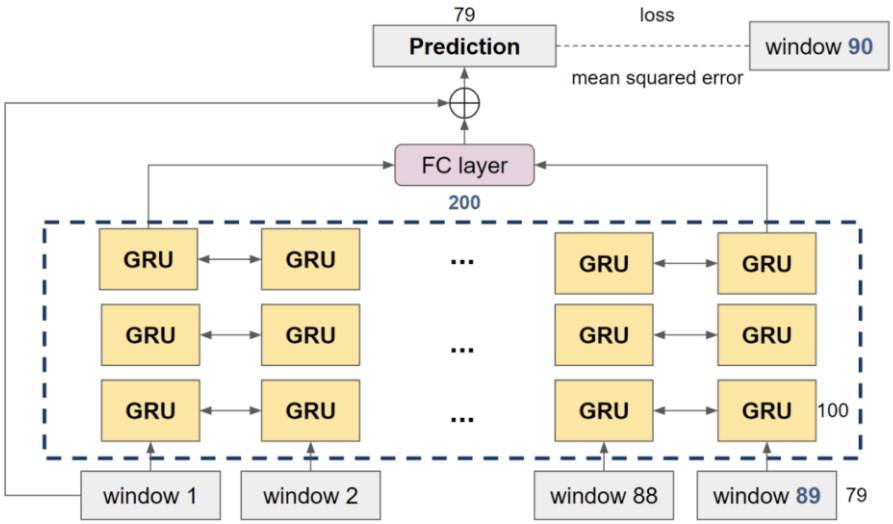

Fig. 3. The model structure uses stacked RNN(GRU) models. For the sliding window, which is the time interval the model trains the specific pattern, set to 90. Using the output of previous 89 data, the model predicts 90th data in the window. The numbers 79 denote the number of features excluding the time feature, 100 denotes the number of hidden cells of GRU, and 200 denotes the number of nodes of the FC (Fully Connected) layer.

On the contrary, a subsequence outlier can be found only by inspecting consecutive instants in time. A subsequence outlier shows a pattern that deviates from the normal repetitive patents as shown in Figure 2. The points between 9 and 10 can be regarded as normal when simply looking at the numerical values, but it is determined as an outlier since its pattern deviated from the repeating patterns between 1 and 2, 5 and 6, 13 and 14, and 17 and 18 [3]. Therefore, it is necessary to detect both outliers for building an intrusion detection system for practical domains. However, most previous studies have focused on detecting point outliers [6, 20].

To address this issue, this paper attempts to detect attacks using multivariate time-series network data. Since time-series network datasets are rarely available, we created a time-series network dataset using the UNSW-NB15 network dataset [7, 13-16]. As an experimental model, we employ an unsupervised approach which contains a stacked RNN model, as was provided by the DACON's HAICon2021 competition [17]. The approach showed a good performance, achieving F1 of 0.926 when the provided code was run on the HAI 2.0 dataset [4]. We carried out preliminary evaluations to test if this approach can be applied to the time-series network data.

2 Model

We use a stacked RNN(GRU) model [5] for learning time-series data in an unsupervised learning manner to detect attacks, which was provided as the baseline model for the HAICon2021 competition. This model uses a three-layer bidirectional

GRU with 100 hidden cells as illustrated in Figure 3. We use the experiment configuration that was set for the baseline model for comparison in the future research. We train the model for 32 epochs keeping the best model parameters, and the parameters that result in the best loss were chosen for evaluation. The window size was set as 90.

3 Time-Series Anomaly Detection Datasets

To evaluate the time-series anomaly detection system we selected two datasets, UNSW-NB15 dataset [7] and HAI 2.0 dataset [4]. The UNSW-NB15 dataset is converted into a time-series format.

3.1 The UNSW-NB15 dataset

The UNSW-NB15 dataset is widely used for benchmarking network intrusion detection systems. The dataset contains 9 network attack behaviors which are Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. The data are provided in two formats, raw traffic packet file and CSV file containing features extracted from captured network flows. We follow Ge et al. [8] to convert the packet data into a time-series format.

Feature Extraction: The raw traffic packets from the UNSW-NB15 dataset were captured using the IXIA PerfectStorm tool and are provided in the PCAP file format [7]. We first select and extract packet fields from the PCAP file using the TShark analyzer tool. Details of the selected fields are shown in Table 1.

Table 1. Detailed information of extracted fields from network packets.

Feature	Field detail
frame	frame.time_epoch, frame.len
ip	ip.src, ip.dst, ip.ttl
tcp	tcp.srcport, tcp.dstport, tcp.stream, tcp.len, tcp.checksum
udp	udp.srcport, udp.dstport, udp.stream, udp.checksum, udp.length

The UNSW-NB15 CSV file contains the flow-based features of labeled flow data. The description of 49 features in the file are listed in Table 2. Each flow is labelled as 0 for normal records and 1 for attacks.

Packet Labelling: After extracting the features from each packet, we sort them in the chronological order using the *frame.time_epoch* feature, which indicates the time information of the packet. The packets in the PCAP file are labelled using the labels in the CSV file. It has information about packets transmitted and a label denoting normal or attack. A label can be created by using the label feature value of the flow which contains the packet.

Table 2. Description of features.

Number	Description	Number	Description	Number	Description
1	srcip	18	Dpkts	35	ackdat
2	sport	19	swin	36	is_sm_ips_ports
3	dstip	20	dwin	37	ct_state_ttl
4	dsport	21	stcpb	38	ct_flw_http_mthd
5	proto	22	dtcpb	39	is_ftp_login
6	state	23	smeansz	40	ct_ftp_cmd
7	dur	24	dmeansz	41	ct_srv_src
8	sbytes	25	trans_depth	42	ct_srv_dst
9	dbytes	26	res_bdy_len	43	ct_dst_ltm
10	sttl	27	Sjit	44	ct_src_ ltm
11	dttl	28	Djit	45	ct_src_dport_ltm
12	sloss	29	Stime	46	ct_dst_sport_ltm
13	dloss	30	Ltime	47	ct_dst_src_ltm
14	service	31	Sintpkt	48	attack_cat
15	Sload	32	Dintpkt	49	Label
16	Dload	33	tcprtt		
17	Spkts	34	synack		

The process of determining whether a particular packet belongs to a flow is as follows. First, *frame.time_epoch* of the PCAP file is matched with the *Stime* value (the 29th field) and the *Ltime* value (the 30th field) of the CSV file. Among the data matched with the packet, we extracted the data that matches the *ip.src* and *ip.dst* of the PCAP with the first field *srcip* and the third field *dstip* of the CSV file. Finally, for TCP, we matched *tcp.srcport* and *tcp.dstport* in the PCAP file, and in the case of UDP, *udp.srcport* and *udp.dstport* in the PCAP file with the 2nd field *sport*, and 4th field *dsport* of the CSV file, and the label of the matched file becomes the label of the corresponding PCAP file. If there is no matching data, it is infeasible to determine whether it is normal or an attack, hence, we removed the corresponding packet. *Tcp* information and *udp* information are integrated into one common information, and then in the case of *ip.src* and *ip.dst*, they are used up to map the PCAP file and the CSV information and then removed. Finally, in the created time-series network data, there are 9 features: *frame.time_epoch*, *frame.len*, *ip.ttl*, *srcport*, *dstport*, *stream*, *checksum*, *len*, and *label*. We removed the label from the data for train, validation and test, since we apply unsupervised learning to dataset, we only used the label for evaluation for validation and test. In total, there are 295,342 time-series data with 277,828 normal data and 17,514 attack data.

Preprocessing: For the source port and destination port features, the port numbers greater than 49,152 are labelled as 2, the numbers greater than 1,024 are labelled to 1,

and the numbers lower than 1,024 are labelled to 0 since they are divided to dynamic port, registered port and well-known port. Then numerical features were scaled to fit 0 to 1 using a min-max scaler.

3.2 **The HAI 2.0 dataset**

The HAI 2.0 dataset is a time-series dataset created for attack detection in cyber-physical systems such as railways, water-treatment, and power plants [4]. The data were collected from the four processes: the boiler process, the turbine process, the water-treatment process, and the HIL simulation. Data samples were collected every second and consist of 80 features. Normal data were collected for 7 continuous days, and the attack data include 38 different attack types. The data are sorted in the increasing order of time feature in the format of "yyyy-MM-dd hh:mm:ss.". Other features contain information associated with the processes such as temperature setpoint, water level setpoint and motor speed.

Preprocessing: To preprocess the data, the timestamp features were dropped, and the numerical features were scaled with a min-max scaler similar to UNSW-NB15 [17]. For some features, of which maximum value and minimum value are the same, we set these features as 0. After scaling features, we applied an exponential weighted function in python function "ewm" with 0.9 for alpha for noise smoothing.

4 Experiments

We compare and analyze the anomaly detection system performance using the UNSW-NB15 and the HAI 2.0 dataset. We convert attack detection into an anomaly detection problem by assuming the attack to be anomalous.

4.1 **Data Preparation**

For both datasets, an unsupervised learning was conducted to train the model using only normal data. We divided the time-series network dataset into training, validation, and test datasets in a ratio of 8:1:1. Then, since the attack data is also included in the training datasets for the time-series network data, we removed attack data in the training datasets. The number of instances for each dataset is presented in Table 3.

Table 3. Simple statistics of processed UNSW-NB15 dataset.

	Training	Validation	Test
Normal	226,240	25,706	25,882
Attack	0	3,828	3,652
Total	226,240	29,534	29,534

However, there are no labels in the test dataset of HAI 2.0 dataset. For the evaluation, we divided the validation dataset, which has labels, into the validation dataset(first 50%)

and the test dataset(last 50%). Table 4 shows the simple statistics of the processed dataset.

Table 4. Simple statistics of the processed HAI 2.0 dataset.

	Training	Validation	Test
Normal	965,603	21,060	21,512
Attack	0	540	89
Total	965,603	21,600	21,601

4.2 **Training**

As described in Figure 3, the model is trained to predict the last sample in the given time window when the preceding samples are given. In order to predict whether the last sample is an anomaly the model is only trained with windows containing normal samples. Theoretically the model will predict the last sample as close as possible to the normal sample given the preceding sample. Therefore, if the difference between the prediction and true last sample is significant, we consider the last sample to be an anomaly. We predict the last sample of the window as an anomaly if the difference is greater than a predetermined threshold. The parameters for training the model are provided in Table 5. The stride means how much data to skip during training.

Table 5. Model parameters and configurations.

parameter	value/name	parameter	value/name
n_hidden	100	n_layers	3
batch_size	512	num_epochs	32
window_size	90	stride	10
loss	MSE	optimizer	AdamW
scheduler	X	dropout	X

4.3 **The Evaluation Metrics**

There are various evaluation metrics such as precision, recall, and *F1* that are frequently used. However, the evaluation metric of time-series data needs to consider various factors such as the diversity of detected attacks and the accuracy of detection as illustrated in Figure 4.

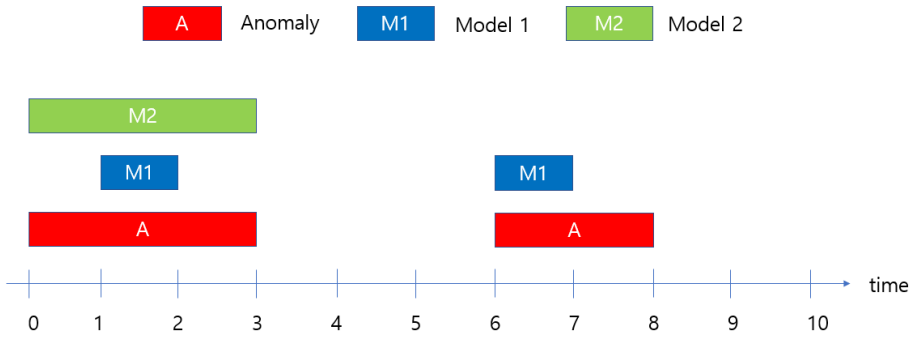

Fig. 4. Illustration of time-series anomaly detection where the two different models Model 1 and Model 2 are used, modified from [18]. The X-axis indicates time, and *A* indicates the time slots where an anomaly exists. *M1* indicates the anomalies that Model 1 detects, and *M2* indicates the anomalies that Model 2 detects.

For example, as shown in Figure 4, Model 2 detects 3 anomaly instances between 0 and 3, and Model 1 detects 2 instances, one between 1 and 2 and the other between 6 and 7. In terms of accuracy, Model 2 outperforms Model 1. However, considering that Model 2 does not detect anomalies between 6 and 8 time slots, it is hard to determine which model performs better. TaPR [19] is an evaluation metric that considers these factors. TaP, which corresponds to precision, is an evaluation metric indicating whether the prediction finds outliers with less false positives. TaR, which corresponds to recall, is an evaluation metric indicating the diversity of the anomalies. Using the detection score TaP^d (resp. TaR^d) and the portion score TaP^p (resp. TaR^p), TaP and TaR can be calculated as follows:

$$TaP = \alpha \times TaP^d + (1 - \alpha) \times TaP^p \quad (1)$$

$$TaR = \alpha \times TaR^d + (1 - \alpha) \times TaR^p \quad (2)$$

where α controls the ratio of TaP^d (resp. TaR^d) and TaP^p (resp. TaR^p), and its value is between 0 and 1 [9].

5 The Experiment Results

This section reports the evaluation results. The figures below show the error and attack distribution of the time-series network data created in this paper and the HAI 2.0 data, respectively.

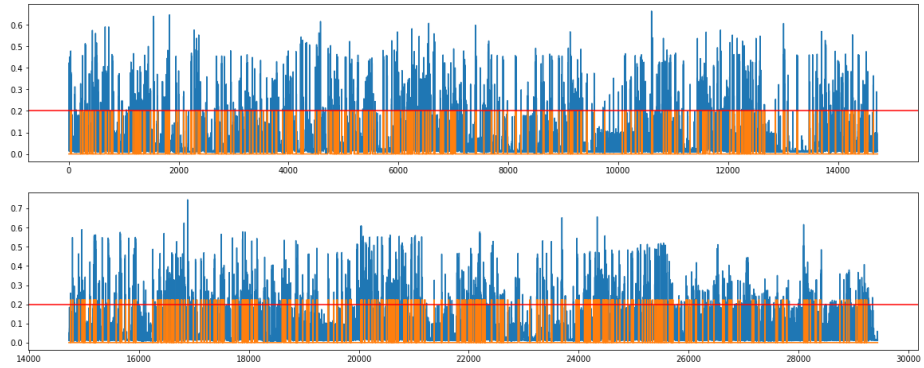

Fig. 5. Distribution of error and attack in validation dataset of the time-series network dataset. The x-axis indicates the order of the data, and the y-axis indicates the absolute difference of (answer - guess). The orange line indicates the attack position, and the blue line indicates the size of the error. The red line is the threshold value that separates the boundary between normal and attack.

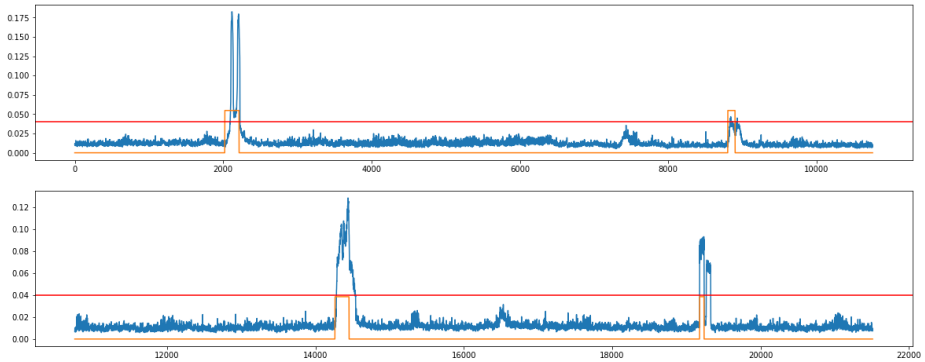

Fig. 6. Distribution of error and attack in validation dataset of the HAI 2.0 dataset. The x-axis indicates the order of the data, and the y-axis indicates the absolute difference of (answer - guess). The orange line indicates the attack position, and the blue line indicates the size of the error. The red line is the threshold value that separates the boundary between normal and attack.

Using the experimental results of validation data, the threshold was set to 0.04 for the HAI 2.0 data, and the threshold was set to 0.2 for time-series network data. The two dataset show different properties. In the HAI 2.0 data, the attack data tends to be greater than the normal data, while in the time-series network data values are relatively evenly distributed. In addition, in the case of the HAI 2.0 dataset, the number of normal data is overwhelmingly larger than that of attack data, unlike the time-series network data. As the evaluation metric, we use TaPR described in Section 3.4. The analyses of the results are shown in the following tables.

Table 6. Detection performance results of UNSW-NB15 data.

Evaluation metric	UNSW-NB15 data
F1	0.737
TaP	0.731
TaR	0.743

Table 7. Detection performance results of HAI 2.0 data.

Evaluation metric	HAI 2.0 data
F1	0.926
TaP	0.861
TaR	1.000

The F1 scores are 0.926 for HAI 2.0 data and 0.737 for time-series network data. The TaP and TaR scores are 0.861 and 1.000 for HAI 2.0 data, and 0.731 and 0.743 for the time-series network data, respectively. This indicates that the model performs better with the HAI 2.0 dataset which contains sensor data.

There are two main factors that account for the poor performance of the time-series network dataset. First, the number of features in the time-series network dataset may be insufficient. In the case of the HAI 2.0 dataset, there are about 80 features, in the case of the time-series network data, only about 10 features were used, making it difficult to determine its anomaly. The other reason is that time-series network data are not complete time-series. In the case of HAI 2.0 dataset, data is generated every second, but in the case of the time-series network dataset, since packets are not transmitted at a specific period, it is difficult to generate data at regular intervals. Moreover, since the attack data is removed from the training data of the time-series network data to learn the normal data only, the time information becomes more irregular.

6 Conclusion

While unsupervised deep learning models have shown great performances in detecting attacks that are point outliers, little has been researched on detecting subsequence outliers. For building a NIDS which can detect subsequence outliers, we first created the time-series network data by processing the UNSW-NB15 dataset. We carried out preliminary experiments using both the HAI 2.0 dataset and the time-series network dataset we created, using a stacked RNN model in an unsupervised manner. The results show that the model performs better with run on the HAI 2.0 dataset than tested on the time-series network dataset. The model achieved F1 scores of 0.926 for the HAI 2.0 data and 0.737 for the time-series network data. The TaP and TaR scores are 0.861 and 1.000 for the HAI 2.0 data, and 0.731 and 0.743 for the time-series network data. The lack of data and insufficient features of the time-series network data can account for its poor performance. We expect that more studies on time-series network data attack detection in the future will help solve these shortcomings.

Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2020-0-00952, Development of 5G Edge Security Technology for Ensuring 5G+ Service Stability and Availability).

References

1. Braei, Mohammad and Sebastian Wagner. “Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art.” ArXiv abs/2004.00433 (2020): n. pag.
2. Bl'azquez-Garc'ia, Ane et al. “A Review on Outlier/Anomaly Detection in Time Series Data.” ACM Computing Surveys (CSUR) 54 (2021): 1 - 33.
3. “Anomaly Detection in Time Series: 2021”, neptune.ai, last modified July 19th, 2021, accessed September 5th, 2021, https://neptune.ai/blog/anomaly-detection-in-time-series.
4. Hyuk-ki Shin, Woomyo Lee, Jeong-Han Yun, and Hyoungchun Kim, "HAI 1.0: HIL-based Augmented ICS Security Dataset", 13th USENIX Workshop on Cyber Security Experimentation and Test, 2020.
5. Cho, Kyunghyun et al. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation.” *EMNLP* (2014).
6. Sandosh, S., V. Govindasamy, and G. Akila. "Enhanced intrusion detection system via agent clustering and classification based on outlier detection." Peer-to-Peer Networking and Applications 13.3 (2020): 1038-1045.
7. Moustafa, Nour, and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)." Military Communications and Information Systems Conference (MilCIS), 2015. IEEE, 2015.
8. Ge, Mengmeng et al. “Deep Learning-Based Intrusion Detection for IoT Networks.” 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC) (2019): 256-25609.
9. Hwang, Won-Seok & Yun, Jeong-Han & Kim, Jonguk & Kim, Hyoung. (2019). Time-Series Aware Precision and Recall for Anomaly Detection: Considering Variety of Detection Result and Addressing Ambiguous Labeling. 2241-2244. 10.1145/3357384.3358118.
10. M. Gupta, J. Gao, C. C. Aggarwal and J. Han, "Outlier Detection for Temporal Data: A Survey," in IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 9, pp. 2250-2267, Sept. 2014, doi: 10.1109/TKDE.2013.184.
11. Song, Y.; Hyun, S.; Cheong, Y.G. A Systematic Approach to Building Autoencoders for Intrusion Detection. In Silicon Valley Cybersecurity Conference, SVCC 2020, San Jose, CA, USA, 17–19 December 2020; Park, Y., Jadav, D., Austin, T., Eds.; Communications in Computer and Information Science; Springer: Cham, Switzerland, 2021; Volume 1383.
12. Song, Youngrok, Sangwon Hyun, and Yun-Gyung Cheong. 2021. "Analysis of Autoencoders for Network Intrusion Detection" Sensors 21, no. 13: 4294. https://doi.org/10.3390/s21134294.
13. Moustafa, Nour, and Jill Slay. "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset." Information Security Journal: A Global Perspective (2016): 1-14.
14. Moustafa, Nour, et al. "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks." IEEE Transactions on Big Data (2017).
15. Moustafa, Nour, et al. "Big data analytics for intrusion detection system: statistical decision-making using finite dirichlet mixture models." Data Analytics and Decision Support for Cybersecurity. Springer, Cham, 2017. 127-156.

16. Sarhan, Mohanad, Siamak Layeghy, Nour Moustafa, and Marius Portmann. NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. In Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings (p. 117). Springer Nature.
17. “HAI DataSet Baseline Model”, DACON, last modified August 2nd, 2021, accessed September 5th, 2021, https://dacon.io/competitions/official/235757/codeshare/3009?page=1&dtype=recent.
18. “[Paper Review] Evaluation Metrics for Time Series Anomaly Detection”, DSBA, last modified September 23th, 2020, accessed September 6th, 2021, http://dsba.korea.ac.kr/seminar/?pageid=3&mod=document&uid=1332.
19. Won-seok Hwang, Jeong-Han Yun, Jonguk Kim, and Hyoungchun Kim, "Time-Series Aware Precision and Recall for Anomaly Detection - Considering Variety of Detection Result and Addressing Ambiguous Labeling", CIKM'19: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019.
20. Devan, Preethi, and Neelu Khare. "An efficient XGBoost–DNN-based classification model for network intrusion detection system." Neural Computing and Applications (2020): 1-16.
21. Malhotra, Pankaj, et al. "LSTM-based encoder-decoder for multi-sensor anomaly detection." arXiv preprint arXiv:1607.00148 (2016).
22. Hundman, Kyle, et al. "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding." Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. 2018.
23. Ding, Nan, et al. "Multivariate-time-series-driven real-time anomaly detection based on bayesian network." Sensors 18.10 (2018): 3367.
24. Park, Daehyung, Yuuna Hoshi, and Charles C. Kemp. "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder." IEEE Robotics and Automation Letters 3.3 (2018): 1544-1551.