Extracting Explanatory Information from LSTM for Binary Classification of Time Series Data for Intrusion Detection

Noriyoshi Ozawa[1] and Shigeki Hagihara[2]

Graduate School of Science and Technology[1],
Faculty of Science and Technology[2],
Chitose Institute of Science and Technology,
758-65 Bibi, Chitose, Hokkaido, 066-8655, Japan
{m2230080, s-hagiha}@photon.chitose.ac.jp

Abstract. In this study, we constructed a method for obtaining information that explains the classification results of a long short-term memory (LSTM) trained as an intrusion detection system (IDS). The LSTM learns two types of information: information about system accesses at each time point and time series information across multiple time points. We extracted explanatory information to rank the importances of these two information types. If the time series information was considered more important, we extracted information indicating which range of past information influenced the classification.

Keywords: Deep learning, explainability, intrusion detection system, XAI

1 INTRODUCTION

Deep learning (DL) can recognize features of complex data without human configuration. These are being applied and improved to adapt to data with various characteristics. In the field of cybersecurity, DL and machine learning (ML) applications have attracted attention because of the recent increase in cyberattacks. Intrusion detection systems (IDSs) that apply DL methods are widely used because they can process large volumes of data logs with high classification accuracy. However, many DL-based IDSs are black box models that cannot provide a basis for their decisions. Thus, there is a need to develop IDSs that can detect an intrusion while simultaneously providing reasons for identifying the intrusion as a threat. For example, if an organization's system experiences a zero-day attack, and the IDS misclassifies the attack, the availability of explanatory information would allow security personnel to rapidly debug and diagnose the system. Many studies have been conducted regarding the development of explainable artificial intelligence (XAI) for IDSs[10]. Some of these studies have included DL models that can learn time series information as an explanatory target. However, insufficient research has been conducted to explain how DL models capture time series information. Cyberattacks are generally classified into two types: attacks with many time series features and attacks with few time series features. For example, denial-of-service (DoS) and probe attacks have time series characteristics because they make many connections to multiple hosts

© The Author(s) 2024
J. Caro et al. (eds.), *Proceedings of the Workshop on Computation: Theory and Practice (WCTP 2023)*, Atlantis Highlights in Computer Sciences 20,
https://doi.org/10.2991/978-94-6463-388-7_12

within a short period of time, whereas remote-to-local (R2L) and user-to-root (U2R) attacks have fewer time series characteristics because the attack contents are embedded within the data. An IDS that learns such data should be able to determine whether to use time series features in its decisions at the time of classification. Therefore, we hypothesized that more appropriate explanatory information could be obtained by determining whether the IDS used time series features when it classified a particular log. In this study, we focused on an LSTM as a DL model that can learn time series information, with the objective of constructing a method for explaining IDS decisions based on time series information. For simplicity, the LSTM used for the IDS assumed a binary model for classifying system accesses as attacks or normal accesses. The proposed method extracts two pieces of information for each classification: the extent to which the LSTM focused on time series information when making its classification decision, and the time period encompassing this focus. The proposed method is described as follows.

1. The classification accuracy of an LSTM trained on network access logs is examined to determine when it does and does not retain memory information. If there is a large difference in classification accuracy between these conditions, time series information is assumed to have significantly contributed to the improvement of classification accuracy.
2. If time series information is considered a significant contributor to LSTM classification accuracy, the number of past logs is increased by one to identify the range of correct decisions to be made.

To confirm the effectiveness of the method, we applied it to LSTMs configured to learn logs in which the time series characteristics of attacks clearly appeared.

The remainder of this article is structured as follows. In Section 2, we provide examples of LSTM applications in intrusion detection and discuss their problems. In Section 3, we explain the position of this research in the field of explainable IDS (XIDS) research. In Section 4, we present the details of the proposed method. In Section 5, we describe the data and the structure of the LSTM used to evaluate the proposed method, then present the LSTM training results. In Section 6, we apply the proposed method to the constructed LSTM and evaluate the results. In Section 7, we discuss the experimental results. In Section 8, we describe related research. In Section 9, we summarize the findings of this study.

2 LSTM APPLICATIONS IN INTRUSION DETECTION AND THEIR PROBLEMS

Muhuri et al. [9] demonstrated that LSTM can be used to construct a highly accurate IDS. We created an IDS using the NSL-KDD dataset[3] with the LSTM configuration presented by Muhuri et al. [9] and performed follow-up experiments. The results showed that DoS, probe, and R2L attacks were classified at probability rates of 0.9981, 0.9889, and 0.8185, respectively; normal accesses were classified at a rate of 0.9948. Although classification accuracy was high, some logs were misclassified. In this case, an IDS consisting solely of LSTM cannot provide decision criteria regarding misclassified logs. Most existing explanatory methods focus on explaining which input features are

considered important and do not utilize time series information. Therefore, such DL explanatory methods cannot be used to accurately explain IDS decisions. To reliably obtain more accurate explanations for DL models that learn time series information, such as LSTM, explanatory methods that utilize this time series information are needed.

3 POSITION OF THIS STUDY IN THE XIDS RESEARCH FIELD

Neupane et al. [10] described the ML methods used for intrusion detection and presented XIDS classification approaches for each method. They determined that explanatory approaches to black box models can be classified as feature-based [8], perturbation-based [12], decomposition-based [13], and hybrid [11]. Feature-based approaches explain the extent to which each feature is involved in the output results. Perturbation-based approaches analyze changes in output caused by perturbations. Decomposition-based approaches primarily focus on the gradient of the model. Despite the differences in these approaches, they share the common goal of assigning a measure of importance to the input space. In this study, we applied a perturbation-based approach after determining whether the model focused on time series information. Thus, in addition to information about which features of the input are important, which is also provided by existing explanatory methods, our approach provides information about which range of past input is important.

4 METHODS FOR EXTRACTING EXPLANATORY INFORMATION FROM LSTM CLASSIFICATION RESULTS

In this section, we propose methods for extracting explanatory information from LSTM classification results. In Section 4.1, we present details of the method used to obtain information regarding the extent to which the LSTM focused on time series information when making classification decisions. In Section 4.2, we demonstrate how to obtain information about the period during which the LSTM focuses on time series information.

4.1 Determination of the Extent to Which the LSTM Focuses on Time Series Information during Classification Decisions

We obtain information about the extent to which LSTM focuses on time series information during classification decisions as follows.

(1) Using the learned LSTM weights, a deep neural network (DNN) is created to perform classification and we calculate the F1 value $F1_{dnn}$. We define a DNN with LSTM weights as a DNN that has no recursive structure, using weights that have already been learned by the LSTM. This DNN is described in detail in Appendix A. The F1 value is a classifier evaluation index for a particular classification problem, where a larger F1 value indicates higher classification accuracy.
(2) Using the learned LSTM weights, classification is performed by the LSTM, and the F1 value $F1_{rnn}$ is calculated.

(3) If $|F1_{dnn} - F1_{rnn}|$ is less than 0.1, the time series information at the time of LSTM classification is considered unimportant. Otherwise, the time series information at the time of LSTM classification is considered important to the classification decision.

In (1) and (2), the difference between the two F1 values is calculated to determine the difference in classification accuracy between networks for which time series information can and cannot be used in classification decisions. In (3), if the difference between the two F1 values exceeds 0.1, the importance of time series information in LSTM classification decisions is considered high.

4.2 Acquisition of Information about the Time Period Focused on by the LSTM

If the above method determines that time series information is important to the classification decision, the following method is proposed to obtain information about the period of interest.

(1) We record rows of data that meet either of the following conditions.

Condition 1
The classification result obtained by a DNN with LSTM weights is a false positive and the result obtained by an LSTM is a true negative.

Condition 2
The classification result obtained by the DNN with LSTM weights is a false negative and the result obtained by the LSTM is a true positive.

For each of the obtained rows, we do the following.

(2) Prepare the data extracted in (1).
(3) Use the data prepared in (2) plus the previous data as an input series for the LSTM.
(4) Provide the input series to the LSTM to calculate the classification probability.
(5) Find the absolute value of the difference between the classification probability obtained in (4) and the classification probability based on all data.
(6) If the absolute value of the difference obtained in (5) is less than or equal to 0.05, the new LSTM input series is regarded as the range of data to be obtained. If the absolute value of the difference is greater than 0.05, the new LSTM input series is regarded as the data series to be input in this iteration plus the previous data series. Return to (4).

In (1), we record the rows of data meeting the indicated conditions to extract data that could not be correctly classified by the DNN using LSTM weights, although they were correctly classified by the LSTM. In (3), (4), (5), and (6), the data series input to LSTM is increased by one row to determine which input data range will result in a classification probability close to the probability that would be obtained if all data were used as the input series. Thus, we obtain information about the period of time focused on by the LSTM.

5 EVALUATION OF THE PROPOSED METHODS

In this section, we evaluate the proposed method by applying it to a trained LSTM. The flow of the method evaluation is described in Section 5.1. In Section 5.2, we describe the data and the structure of the LSTM used in the experiment, then determine the classification accuracy of the LSTM network after training.

5.1 Evaluation Methods

An LSTM with high classification accuracy appropriately captures and classifies the characteristics of the data and its appearance patterns. Thus, if the explanatory information obtained by applying the proposed method to LSTMs trained on data with clear time series characteristics represents the time series characteristics of the data, we conclude that the proposed method correctly extracted explanatory information. Therefore, we created data with clear time series characteristics and appearance patterns to evaluate the accuracy of the proposed method. We also defined the importances of the time series features, allowing clarification of which data had time series characteristics that were important criteria for classification decisions. We evaluated the method used to obtain information about the extent to which the LSTM focused on time series information for classification decisions, as described below. If the method determines that an NN focused on time series information at the time of classification, it also determines whether the time series features of the training data had a high level of importance. We evaluated the method used to obtain information about the period of time during which an LSTM focused on time series data by examining whether an LSTM trained on data with important time series features yielded results that required a larger range of data for accurate classification.

5.2 Construction of the DL Network for Experiments

In this section, we describe the data used in our experiments, the structure of the LSTM, and the LSTM training process.

Data creation Because our proposed method is intended for application to LSTM networks used as IDSs, we created data representing the characteristics of actual cyberattacks, with reference to cyberattacks included in the NSL-KDD dataset[3]. According to the UCL Machine Learning Repository[4], which publishes the KDD Cup '99 dataset that preceded the NSL-KDD dataset, DoS and probe attacks have time series characteristics because they make many connections to multiple hosts within a short period of time. Conversely, R2L and U2R attacks do not have time series characteristics because the attack contents are embedded in the data. Therefore, we created three types of data: data with large features, imitating R2L and U2R attacks (Data 1); data with time series features and occurrence patterns, which imitate DoS and probe attacks (Data 2); and data containing only time series features (Data 3). The third dataset was created to confirm that data comprising purely time series features can be used to obtain time series explanations. The datasets are described in greater detail in Table 1.

Table 1. Features of the three datasets used in experiments.

Dataset	Assumed cyberattacks	Data features	Time series features
Data 1	R2L and U2R attacks	◎	×
Data 2	DoS and probe attacks	○	○
Data 3	Time series features	×	◎

The contents of each row of each dataset are shown in Table 2. Each row is represented by a 16-digit binary number, where a hyphen indicates that the number was randomly chosen from 0 or 1. Rows added to represent normal system access logs consisted of numbers with all digits randomly chosen from 0 or 1, without time series characteristics. Rows added to represent attack logs consisted of either large or small amounts of time series characteristics. This design allowed the use of two networks to evaluate the proposed method: one that does not focus on time series features in the data occurrence patterns and one that slightly focuses on these patterns. Rows of attacks were identified as attacks in which 1s and 0s are recorded alternately (e.g., row 1 in Table 2) and attacks in which the last four digits from the left are 0s; the 7th, 10th, and 13th digits are 1s; and the remaining digits are randomly chosen from 0 or 1 (e.g., row 2 in Table 2).

Table 2. Contents of each row.

Row name	Description	Data
Normal row	Random	- - - - - - - - - - - - - - - -
Attack row 1	1s and 0s appear alternately	1010101010101010
Attack row 2	Fixed number of specific digits	0000 - - 1 - - 1 - - 1 - - -

In the construction of Data 1, an attack row is added with a probability of 0.25 and a normal row is added with a probability of 0.75. Thus, attack rows 1 and 2 shown in 2 are regarded as Data 1-1 and Data 1-2, respectively. In the construction of Data 2, two states are considered: normal and attack. In the normal state, a normal row is added with a probability of $1 - 0.05$; a transition to the attack state occurs at a probability of 0.05. In the attack state, an attack row is added with a probability of 0.9; a normal row is added with a probability of $1 - 0.9$. After the attack row is added with a probability of 0.9 for 16 rows, it always returns to the normal state. Attack rows 1 and 2, along with a random case, are regarded as Data 2-1, Data 2-2, and Data 2-3, respectively. In the construction of Data 3, a sine wave is added in every cycle; each cycle changes its period or adds noise with a probability of 0.3. These datasets are described in greater detail in Appendix B.

Determination of time series feature importance When defining the importance of time series features, the data features and time series features are expressed as numerical scores. Concerning the data characteristics, data using row 1 of the attack are given a score of 3, data using row 2 of the attack are given a score of 2, data with a randomly

selected attack row are given a score of 1, and data constructed using a sine wave are given a score of 0 because these data are meaningless. Concerning the time series characteristics, data with intermittent attacks are given a score of 0 because they have no time series information, data with intensive attacks are given a score of 2, and data constructed using a sine wave are given a score of 3. The difference between the score of the time series feature and the score of the data feature is regarded as the importance of the time series feature for each datum. Data with positive importance values for time series features are assumed to have important time series features. The characteristics of each dataset are summarized in Table 3, where scores of 3, 2, 1, and 0 are indicated by ◎, ◯, △, and ×, respectively.

Table 3. Data features for each row of the three datasets.

Data row	Data features	Time series features	Importance of Time series features
Data 1-1	◎	×	-3
Data 1-2	◯	×	-2
Data 2-1	◎	◯	-1
Data 2-2	◯	◯	0
Data 2-3	△	◯	1
Data 3	×	◎	3

LSTM configuration and training The hyperparameters of the LSTM are shown in Table 4. The learning rate was changed to 0.001 at 50% of the epoch and then to 0.0005 at 75% of the epoch; its initial value of 0.01 is desirable in most cases, and reduction of the learning rate is undesirable as learning progresses [1]. The dropout rate was set to 0.01 and the activation function was set to a sigmoid function. Optimization methods were trained using the stochastic gradient descent (SGD) and Adam optimizers, respectively; the optimization method that showed adequate learning progress was selected. The number of epochs was set to 10. The batch size was set to 1 because LSTM constructed using the tf.Keras module requires training and testing batch sizes with similar lengths; thus, the batch size during training must be set to 1 to match the batch sizes used in testing and application. The loss function was the cross-entropy error function, which is used in binary classification; the number of neurons was set to 50, and the number of layers was set to 3 to achieve DL.

Table 4. Hyperparameters of the long short-term memory (LSTM) method. SGD, stochastic gradient descent.

Hyperparameter	Set value
Learning rate	0.01, 0.001, 0.0005
Dropout rate	0.01
Activation function	Sigmoid
Optimizer	Adam, SGD (In Data 2-3 and 3)
Epochs	10
Batch size	1
Loss function	Cross entropy error
Number of neurons	50
Number of layers	3

The LSTM was trained with each dataset created as input. The first 80% of each dataset was used as training data, and the remaining 20% was used as test data. The accuracy of the data is shown in Table 5. The index attached to the network name represents the index of the trained data. The trained LSTM was able to classify all data with high accuracy.

Table 5. Data accuracy and confusion matrix values.

Network name	Accuracy	Precision	Recall	F1 value
Network 1-1	1.00	1.00	1.00	1.00
Network 1-2	0.98	0.99	1	0.99
Network 2-1	1.00	1.00	1	1.00
Network 2-2	0.99	0.99	1.00	0.99
Network 2-3	0.99	0.91	0.79	0.88
Network 3	0.78	0.89	1	0.88

6 EVALUATION OF THE PROPOSED METHOD

In this section, we present results obtained through application of the proposed method to the LSTM networks described in Section 5, then evaluate the results.

6.1 Method for Obtaining Information about the Extent to Which the LSTM Focused on Time Series Information during Classification Decisions

Experimental results The value of $F1_{rnn}$ for LSTM classification was compared with the value of $F1_{dnn}$ for DNN classification using LSTM weights. The results are shown in Table 6.

Table 6. F1 value comparison between LSTM classification and DNN classification using LSTM weights.

Network	$F1_{dnn}$	$F1_{rnn}$	$\lvert F1_{rnn} - F1_{dnn} \rvert$
Network 1-1	0.97	1.00	0.03
Network 1-2	0.89	0.99	0.10
Network 2-1	0.98	1.00	0.02
Network 2-2	0.90	0.99	0.09
Network 2-3	0.11	0.88	0.77
Network 3	0.07	0.88	0.81

Table 4 shows the results of judging the importance of time series information during classification according to the difference in $F1$ values 7.

Table 7. Importance judgment results for each method.

Network	Importance judgment
Network 1-1 Network 1-2 Network 2-1 Network 2-2	Time series information is unimportant
Network 2-3 Network 3	Time series information is important

Evaluation Next, we investigated whether information obtained by applying the proposed method to the trained LSTM network was able to classify LSTM networks trained on data with time series features in the data occurrence patterns and LSTM networks trained on data without time series features. The judgment results obtained by applying the method (Table 7) are mapped to the importance of time series information (Table 3) in the trained data in Table 8. These results show that a network trained on data with positive importance values for time series features requires a time series to classify an attack/non-attack. Thus, the proposed method can be applied to trained LSTM networks to determine the importance of memory information in LSTM classification.

Table 8. Judgment results for each method.

Network	Judgment result	Importance of time series features
Network 1-1	Time series characteristics not required	-3
Network 1-2		-2
Network 2-1		-1
Network 2-2		0
Network 2-3	Time series characteristics required	1
Network 3		3

6.2 Method for Obtaining Information about the Time Period Focused on by the LSTM

Experimental results The proposed method was applied to an LSTM network trained on each dataset. The mean, standard deviation, minimum, first quartile, median, third quartile, and maximum values of the obtained ranges are summarized in Tables 9 and 10 for all data that met conditions 1 and 2, respectively. Values were rounded to the third decimal place.

Table 9. Statistical results for the range extracted from data meeting condition 1. Max, maximum; Min, minimum; SD, standard deviation.

Dataset	Number of datasets	Mean	SD	Min	First quartile	Median	Third quartile	Max
Data 1-1	310	2.20	0.54	2	2	2	2	5
Data 1-2	1090	2.30	0.48	2	2	2	3	4
Data 2-1	294	2.30	1.10	2	2	2	2	19
Data 2-2	1648	2.18	0.40	2	2	2	2	5
Data 2-3	730	3.58	2.61	2	2	3	3	25
Data 3	25	10.72	1.21	10	10	11	11	16

Table 10. Statistical results for the range extracted from data meeting condition 2.

Dataset	Number of datasets	Mean	SD	Min	First quartile	Median	Third quartile	Max
Data 1-1	0	-	-	-	-	-	-	-
Data 1-2	0	-	-	-	-	-	-	-
Data 2-1	0	-	-	-	-	-	-	-
Data 2-2	0	-	-	-	-	-	-	-
Data 2-3	5917	5.51	4.37	2	3	4	6	38
Data 3	793	15.13	4.32	7	12	15	18	24

Evaluation of results To evaluate the proposed method, we attempted to obtain results in which networks trained on data with more important time series features require a larger range of data for accurate classification. We selected the third quartile of each measure as the size of the range of input series that would allow each network to accurately classify each piece of data. Data that met condition 1 are data that correctly classified normal data after the addition of time series information. The third quartile column of Table 9 shows that data with a time series feature importance value of 0 or less can be correctly classified using approximately 3 rows of data. In contrast, data with positive importance values require a larger range of data; thus, Data 2-3 and 3 were accurately classified using approximately 4 and 11 rows of data, respectively. This result is consistent with the importance of time series characteristics. Data that met condition 2 are data that correctly classified attack data after the addition of time series information. As shown in Table 10, there were no data meeting condition 2 for which the importance

of time series information was less than 0. This result is reasonable considering that the attack data were accurately classified even without time series information; those data contained significant characteristics. In contrast, some data that met condition 2 had positive importance values for time-series features. The range of the extracted series was larger for Data 3 than for Data 2-3. Because the importance values of time series features were greater in Data 3 than in Data 2-3, this result is consistent with the importance of time series features. Overall, these results confirm that the proposed method can be used to obtain correct information.

7 DISCUSSION

In this section, we discuss and interpret the evaluation results for the proposed method. In Section 7.1, we discuss the method used to obtain information about the extent to which the LSTM focused on time series information during classification decisions. In Section 7.2, we discuss information obtained through our evaluation of the method. In Section 7.3, we discuss information obtained regarding the time period focused on by the LSTM. In Section 7.4, we discuss challenges involved in applying the proposed method to IDS.

7.1 Methods for Obtaining Information about the Extent to Which the LSTM Focused on Time Series Information during Classification Decisions

In this study, a threshold of 0.1 was specified for the difference in $F1$ values used to determine whether time series information was focused on by the LSTM. Because the network classification results were appropriate (Table 7), we conclude that this threshold was appropriate for the data utilized in this study. However, to determine the most appropriate threshold, exhaustive examination is needed concerning the effects of threshold alteration on the resulting accuracy. In the proposed method, if the $F1$ difference is below the threshold, the time series information is considered unimportant; however, if some data can be successfully classified by adding time series information, it is meaningful to explain the results obtained by learning the time series information. The threshold should be regarded as a guide, such that even if the network is below the threshold value, a comprehensive judgment should be made via the proposed method if time series information can be obtained. Although the $F1$ value was used for evaluation in this study, this choice depends on the characteristics of the data to be classified. For example, when every attack row is classified as an attack, the recall ratio should be used for evaluation.

7.2 Evaluation of Methods Used to Obtain Information about the Extent to Which the LSTM Focused on Time Series during Classification Decisions

In this section, we summarize and discuss information obtained by comparing the LSTM classification results with the classification results for DNN using LSTM weights. We calculated the change in each value of a confusion matrix obtained through LSTM classification by LSTM and by DNN using the LSTM weights (Table 11). When the network

was trained on data that contained large features, true positive rows did not increase, whereas true negative rows increased when time series information was considered during classification. Thus, for data that already show high classification accuracy using a DNN with LSTM weights, the use of LSTM may lead to fewer false positives. Some networks also showed a decrease in the number of true positives or true negatives during LSTM classification, indicating that for some data, the addition of time series information reduces the accuracy of the classification decision. Therefore, we conclude that the improvement in classification accuracy achieved by switching from DNN-based classification using LSTM weights to LSTM-based classification is not simply the result of additional time series information. Future research should examine why the inclusion of time series information resulted in incorrect classification results.

Table 11. Comparison of confusion matrix values.

Network	True positive	True negative	False positive	False negative
Network 1-1	-16	+310	-310	+16
Network 1-2	0	+1090	-1090	0
Network 2-1	0	+295	-295	0
Network 2-2	-1	+1648	-1648	+1
Network 2-3	+5848	+686	-686	-5848
Network 3	+771	-172	+172	-771

The proposed method focused on the process by which LSTMs learn time series information from input data, thereby investigating the influence of time series information on LSTM classification results. Some DL networks are able to learn time series information, such as [5], which is a modified LSTM, and Gated Recurrent Unit (GRU)[2]. The proposed method may also be applicable to these DL networks.

7.3 Method for Obtaining Information about the Time Period Focused on by the LSTM

In this study, the absolute value of the difference in classification probabilities was set to a threshold of 0.05, which allowed determination of the time period focused on by the LSTM. Because information extracted during the method evaluation was considered appropriate, as described in Section 6.2, we conclude that a threshold value of 0.05 is appropriate for the datasets created in this study. However, to determine the optimal threshold, the effects of threshold alteration on the accuracy of the method should be exhaustively tested. The proposed method has some limitations. It is based on past information that focuses on the shortest range of data that meets each condition. For an LSTM trained on data requiring long, complex memory, the classification probability may decrease when a longer series is classified as input; it may increase again when an even longer series is input. Because the proposed method does not consider such cases, it may be unable to extract specific data ranges. The identification of such ranges is a topic for future research.

7.4 Challenges in IDS Applications

When applying the proposed method to IDS at runtime, it is important to maintain a high processing speed. To obtain insights into the processing time of the proposed method, we measured the time required to calculate all data in Data 3 that met condition 2, using a method for obtaining information about the time period focused on by the LSTM. Approximately 636 s of processing time were required for 793 rows of data. Thus, if all rows were processed at the same rate, each row would require approximately 0.8 s of processing time. For actual IDS applications, it is unrealistic to require 0.8 s of processing time each time a system access is considered an attack; these accesses occur on a continual basis. Therefore, it is preferable to construct a system in which new calculations are performed in parallel. If the reduction of execution time is more important than the accuracy of the method, it is effective to increase the range for calculation by two or three rows, rather than increasing the range by one row as in the proposed method.

8 RELATED RESEARCH

Many previous studies have utilized DL for IDS, including Muhuri et al. work[9], in which an IDS was run using LSTM with a genetic algorithm (GA) for feature selection. The LSTM was used to classify the types of attacks, and the results were compared with the output of several other ML models; the classification accuracy of LSTM using the GA model was far superior to the other tested models. The use of Transformer as an IDS was also recently explored [7]. Research is also underway concerning other XAI approaches for DL models. Ying et al. [14] proposed a method to explain decisions made by graph neural networks (GNNs); the method was able to indicate important graph structures and nodes with important predictive roles in the model. Additionally, there are several LSTM variants, including one with a peephole connection [5] and GRU [2].

9 CONCLUSION

In this study, we explored the application of LSTM as a DL model that can learn time series information to construct and evaluate an explanatory method for the inclusion of time series information in system access log classification. The method was evaluated using LSTM networks trained on data with obvious time series characteristics to obtain information regarding the extent to which the LSTM focused on time series information during classification decisions and to obtain information about time periods focused on by the LSTM. We defined the importance of time series features in the created datasets and examined whether information obtained by the model after training was consistent with the actual importance of the time series features. The evaluation results showed that the proposed method is effective. The findings of this study represent a first step toward the identification of decision criteria for DL networks that can learn time series data and provide a foundation for configuring XIDSs with high accuracy, enabling security personnel to accurately and quickly respond to cyberattacks.

REFERENCES

1. Bengio, Y.: Practical recommendations for gradient-based training of deep architectures. CoRR abs/1206.5533 (2012), http://arxiv.org/abs/1206.5533
2. Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. CoRR abs/1406.1078 (2014), http://arxiv.org/abs/1406.1078
3. for Cybersecurity, C.I.: NSL-KDD dataset. https://www.unb.ca/cic/datasets/nsl.html, Accessed: 2023-02-01
4. Dua, D., Graff, C.: UCI machine learning repository. http://archive.ics.uci.edu/ml (2017), Accessed: 2023-02-01
5. Gers, F., Schmidhuber, J.: Recurrent nets that time and count. In: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium. vol. 3, pp. 189–194 vol.3 (2000)
6. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation 9(8), 1735–1780 (1997)
7. Li, Y., Yuan, X., Li, W.: An extreme semi-supervised framework based on transformer for network intrusion detection. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. p. 4204–4208. CIKM '22, Association for Computing Machinery, New York, NY, USA (2022), https://doi.org/10.1145/3511808.3557549
8. Lundberg, S., Lee, S.I.: A unified approach to interpreting model predictions (2017)
9. Muhuri, P.S., Chatterjee, P., Yuan, X., Roy, K., Esterline, A.C.: Using a long short-term memory recurrent neural network (lstm-rnn) to classify network attacks. Inf. 11, 243 (2020)
10. Neupane, S., Ables, J., Anderson, W., Mittal, S., Rahimi, S., Banicescu, I., Seale, M.: Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities. IEEE Access 10, 112392–112415 (2022)
11. Pang, G., Ding, C., Shen, C., van den Hengel, A.: Explainable deep few-shot anomaly detection with deviation networks (2021)
12. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should I trust you?": Explaining the predictions of any classifier. CoRR abs/1602.04938 (2016), http://arxiv.org/abs/1602.04938
13. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-CAM: Visual explanations from deep networks via gradient-based localization. International Journal of Computer Vision 128(2), 336–359 (oct 2019), https://doi.org/10.1007%2Fs11263-019-01228-7
14. Ying, R., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: GNN explainer: A tool for post-hoc explanation of graph neural networks. CoRR abs/1903.03894 (2019), http://arxiv.org/abs/1903.03894

A LSTM AND DNN USING LSTM WEIGHTS

LSTM

LSTM is a network proposed by Hochreiter[6] that uses recurrent neural network (RNN) units with a modified structure capable of handling long-term memory. The calculation of the LSTM unit at time t uses the hidden state h_{t-1}, comprising the output from the LSTM unit at the previous time t_1, plus the value of the memory cell c_t, which is prepared to hold past memory. The input gate i and output gate o, with unique weights w_i, w_o, respectively, are introduced for calculations of the input to and output from the LSTM unit, respectively, to learn how much past and current information should be retained. These improvements allow proper learning of dependencies between long-term data. Equation (1) is used to calculate the output of the LSTM unit, where t is the time, $\mathbf{x}$ is the input vector, $\mathbf{W}$ is the weight matrix, and b is the bias. If f is the output of the forgetting gate, g is the output of the acquiring gate, which obtains the newly remembered content; i is the output of the input gate, o is the output of the output gate; c is the output of the memory gate; and h_t is the output from the LSTM unit. These values are calculated using the following formula.

$$\begin{aligned} f_t &= \sigma(\mathbf{x}_t \mathbf{W}_{\mathbf{x}f} + h_{t-1}\mathbf{W}_{\mathbf{h}f} + b_f) \\ g_t &= \tanh(\mathbf{x}_t \mathbf{W}_{\mathbf{x}g} + h_{t-1}\mathbf{W}_{\mathbf{h}g} + b_g) \\ i_t &= \sigma(\mathbf{x}_t \mathbf{W}_{\mathbf{x}i} + h_{t-1}\mathbf{W}_{\mathbf{h}i} + b_i) \\ o_t &= \sigma(\mathbf{x}_t \mathbf{W}_{\mathbf{x}o} + h_{t-1}\mathbf{W}_{\mathbf{h}o} + b_o) \\ c_t &= f_t \odot c_{t-1} + g_t \odot i_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{1}$$

The $\odot$ denotes the element-by-element product of the matrix. Note that the input $\mathbf{x}_t^k$ to the LSTM cell in the kth layer ($k = 1, 2, \ldots, K$) when the LSTM is multilayered into K layers is the output h_t^k from one previous layer. When training data are input into the LSTM network using the LSTM cell composed above, and learning is conducted such that the error between the output from the network and the correct answer is minimized, the weights $\mathbf{W}_\mathbf{x}$ and $\mathbf{W}_h$ are adjusted to appropriately represent the overall data; the time series information of the data are adjusted to appropriately represent the overall data.

DNN with LSTM weights

In this study, we used LSTM without recursive processing, which we termed a DNN with LSTM weights. The formula used to calculate the output from the neuron of a DNN with LSTM weights is equivalent to Equation 1 when $h_{t-1} = 0$ and $c_{t-1} = 0$. Because this approach does not include memory information, the network does not have a forgetting gate. The output from a DNN neuron using LSTM weights is calculated as

follows.

$$
\begin{aligned}
g_t &= \tanh(\mathbf{x}_t \mathbf{W}_{\mathbf{x}g} + b_g) \\
i_t &= \sigma(\mathbf{x}_t \mathbf{W}_{\mathbf{x}i} + b_i) \\
o_t &= \sigma(\mathbf{x}_t \mathbf{W}_{\mathbf{x}o} + b_o) \\
c_t &= g_t \odot i_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
\tag{2}
$$

B DEFINITION OF TRAINING DATA

In this section, we describe training data used in the evaluation.

Definition of Data 1 Each state that creates a row in Data 1 is defined as follows.

- $S_0 \cdots$ Initial state.
- $S_{n_1} \cdots$ Normal state. Add one normal row.
- $S_{a_1} \cdots$ Attack state. Add one attack row.

The probability model for Data 1 is shown in Figure 1. From the initial state, it transitions to state S_{a_1} with a probability of 0.25 and to state S_{n_1} with a probability of $1 - 0.25$. Subsequently, it continues to transition from each state to state S_{a_1} with a probability of 0.25 and to state S_{n_1} with a probability of $1 - 0.25$. Thus, Data 1 comprises data generated according to the rule that an attack row is added with a probability of 0.25 and a normal row is added with a probability of 0.75. For each type of attack row, Data 1-1 and Data 1-2 were created within Data 1. Attack rows added to Data 1-1 were designated attack row 1, and attack rows added to Data 1-2 were designated attack row 2. The attack rows added to each dataset are shown in Table 12.

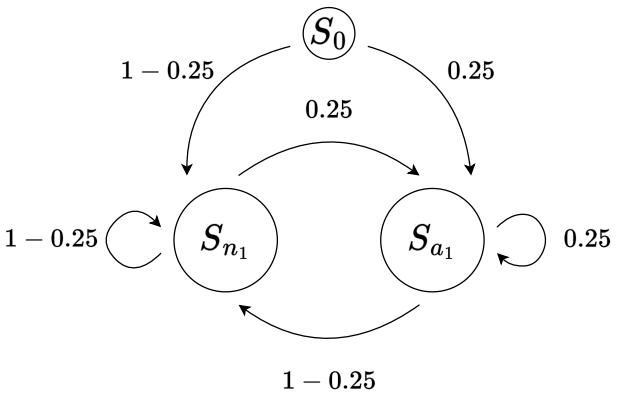

Fig. 1. Probability model for Data 1.

Table 12. Characteristics of each dataset

Dataset	Row of attack
Data 1-1	1010101010101010
Data 1-2	0000 - - 1 - - 1 - - 1 - - -

Definition of Data 2 Each state that creates rows in Data 2 is defined as follows.

- $S_0 \cdots$ Initial state.
- $S_{n_2} \cdots$ Normal state. Add one normal row.
- $S_{a_2} \cdots$ Attack state. Continue until 16 rows are added. During this state, an attack row is added with a probability of 0.9 and a normal row is added with a probability of 0.1.

The probability model for Data 2 is shown in Figure 2. From the initial state, it transitions to state S_{a_2} with a probability of 0.05 and to state S_{n_2} with a probability of $1 - 0.05$. Subsequently, it transitions from state S_{n_2} to state S_{a_2} with a probability of 0.05; from state S_{a_2}, it always transitions to state S_{n_2}. State S_{a_2} continues until 16 rows of data are added, during which an attack row is added with a probability of 0.9 and a normal row is added with a probability of $1 - 0.9$. Thus, Data 2 is generated according to the rule that it will transition to the attack state with a probability of 0.05 and will always return to the normal state after the attack row is added, with a probability of 0.9 for 16 rows. Within Data 2, the data were defined by the type of attack row, creating Data 2-1, Data 2-2, and Data 2-3. Attack rows added to Data 2-1 were designated Attack Row 1, and attack rows added to Data 2-2 were designated Attack Row 2. In Data 2-3, a random 16-digit binary number is generated when the state changes to S_{a_2}, and a line of attack is processed as a line of attack in state S_{a_2}. The attack rows added to each dataset are shown in Table 13.

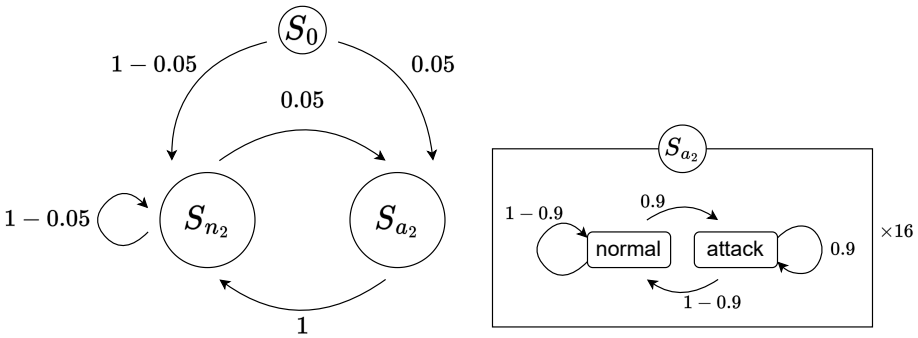

Fig. 2. Probability model for Data 2.

Table 13. Characteristics of each dataset

Data	Row of attack	p_{a_1}
Data 2-1	1010101010101010	0.05
Data 2-2	0000 - - 1 - - 1 - - 1 - - -	0.05
Data 2-3	- - - - - - - - - - - - - - - -	0.05

Definition of Data 3 Each state that creates rows in Data 3 is defined as follows.

- $S_0 \cdots$ Initial state.
- $S_{\sin 2\pi} \cdots$ Normal state. Add the value of y in $y = 1 + \sin \frac{2\pi x}{100}$ with 100 rows per period, converted to 16-digit binary numbers, as rows.
- $S_{\sin 8\pi} \cdots$ Attack state. Add the value of y in $y = 1 + \sin \frac{8\pi x}{100}$ with 25 rows per period, converted to 16-digit binary numbers, as rows.
- $S_{noise} \cdots$ Noise generating state. Add a row with a randomly chosen decimal number in the range [0, 2] converted to a 16-digit binary number.

The probability model for Data 3 is shown in Figure 3. From the initial state, there is always a transition to state $S_{\sin 2\pi}$. From state $S_{\sin 2\pi}$, it transitions to state $S_{\sin 8\pi}$ with a probability of 0.5; from state $S_{\sin 8\pi}$, it transitions to state $S_{\sin 2\pi}$ with a probability of 0.99. From state $S_{\sin 2\pi}$ or state $S_{\sin 8\pi}$, it transitions to state S_{noise} with a probability of 0.3; from state S_{noise}, it transitions to state $S_{\sin 2\pi}$ or state $S_{\sin 8\ pi}$ with a probability of $1 - 0.3$. Thus, Data 3 is generated according to the rule that sin waves are added every cycle, and the period of the sin wave is changed every cycle or noise is added with a probability of 0.3.

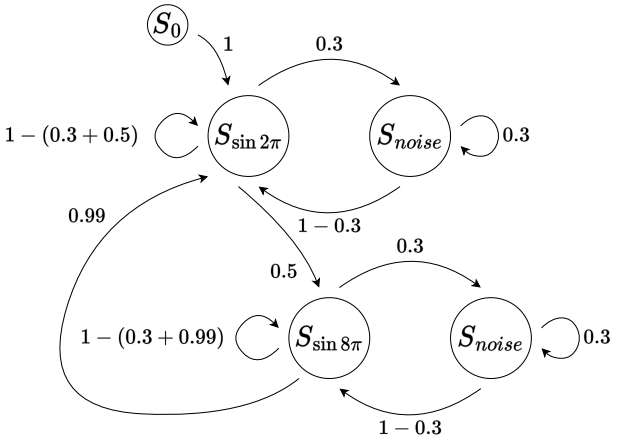

Fig. 3. Probability model for Data 3.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (http://creativecommons.org/licenses/by-nc/4.0/), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.