sBiLSAN:stacked Bidirectional Self-Attention LSTM Network for Anomaly Detection and Diagnosis from System Logs

Chenyu You
Department of Electrical Engineering
Stanford University
uniycy@stanford.edu

Qiwen Wang
Department of Computer Science
Stanford University
qwang26@stanford.edu

Chao Sun
Department of Computer Science
Stanford University
chao.sun@stanford.edu

Abstract

High service availability is crucial for computer systems. Monitoring computing systems has become increasingly difficult as researcher and system analysts face the challenge of analysis a wide range of monitoring information. Thus, the anomaly detection system along with firewalls and intrusion prevention systems are the must-have tools. The primary purpose of a system log is to record system states and significant events for enhanced system reliability. Such system logs are universally available in all computer system. Efficient anomaly detection and prediction via log mining over unstructured texts are highly required. Therefore, we seek to leverage machine learning (ML)-based model to increase the reliability of computer systems. This work aims to detect and predict system failures and errors via stacked Bidirectional self-attention long short-term memory (LSTM) networks in certain time intervals. Also, we present a comprehensive study and evaluation of existing anomaly detection algorithms, using a new large-scale benchmark consisting of both synthetic and real-world network traffic failures of various error types. Our evaluation and analysis indicate the performance of our proposed method can capture the complex representations of the anomaly, and obtain promising results as compared to the other state-of-the-art methods.

1. Introduction

System log analysis is critical for a wide range of tasks in maintaining large-scale computer systems such as high-performance computing clusters and cloud computing. Log files, generated in almost all computer systems today, contain highly valuable information about health and behavior of the system and thus they can be utilized to analyze behavioral aspects of the system. However, there exists a large number of log entries in computer systems; it is challenging to seek relevant information in these files. Thus, computer-based log analysis techniques are indispensable for the process of finding relevant data in log files. The major issue to detect the cause of errors is the lack of enough information by searching with regular expressions. For the aspect of computer networks, a typical computer system encompasses a wide range of issues, such as intrusion detection, preventative security, and network monitoring, and associated investigative and remediation efforts [15]. Network errors are one of the most significant reasons that result in service unavailability. Network errors can be seen as a form of gray failure, *e.g.*, sector error and latency error, since subtle failures are hard to be detected, even when applications are afflicted by them.

These consist of critical security tasks such as intrusion detection, malware detection, maintenance tasks, and modeling data or traffic flow patterns. Since computer systems and applications become more complicated than before, they are subject to more bugs and vulnerabilities for online attacks. Also, the attacks are becoming increasingly more sophisticated, and most users only realize they are under attack when attackers have leaked data and corporate secrets. Extracting knowledge from a wide range of system logs is complicated by several factors:

1. The system generates log files including over terabytes of data per day.

2. Label data applied in the current investigation are system specific and biased.

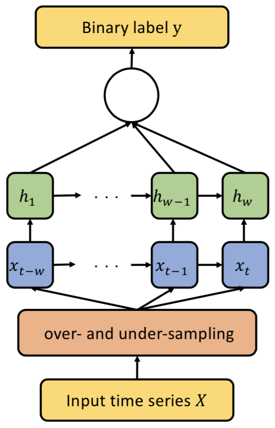


Figure 1. A visualization of LSTM network.

3. The connections across logging sources and system entities such as processes, nodes, and I/O, are obscure and complex.

According to these factors, challenges central to anomaly detection in multivariate time series data hold for the network system. Data being monitored are often heterogeneous, noisy, and high-dimensional. In scenarios, anomaly detection is being used as a diagnostic tool for identifying the existence of a potential issue on the network. Also, the recent work reports that it is challenging to diagnose a problem using unstructured logs even after the error location has been determined [30]. Many traditional anomaly detection methods and unaided human monitoring are no longer effective. The reasons can be identified in two folds. First, traditional methods often focus on the low-dimensional domain and will face difficulties when fitting the high-dimensional data. Second, they require the manual engineering of features [13]. Therefore, how to derive the highly effective automated methods for visualization and analysis of system logs has been one of the major endeavors in the computer security field. The ideal system should be able to detect attacks, diagnose attacks, and deploy countermeasures and repairs [21].

Almost all computer systems document the history of events and states in system logs. Such operation serves as a keyhole to computer system work-flow and allows maintenance of large-scale systems for either business or computation purposes. Access to the past processes from logs also provides an approach for monitoring and anomaly detection. Current methods of anomaly detection using log data can be categorized into workflow analysis, Principal component analysis (PCA) of message counters, and invariant mining of co-occurrence patterns, each of which can be restricted in different cases to defend online attacks.

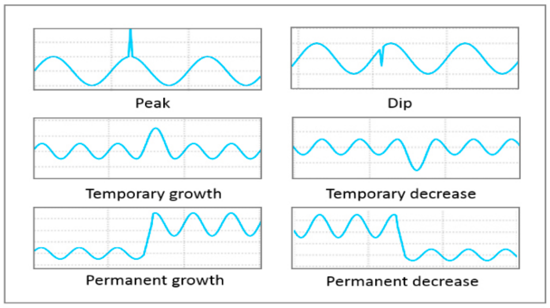


Figure 2. The different anomaly types.

Challenges vary from deviation of log formats in different systems, to problem identification with the detected errors, and to the use of extensive data in online anomaly detection. Although the rule-based methods are proposed, the prerequisite to extract features in the logs deviates it from general purpose [30, 32]. To intervene attack currently in progress, prompt anomaly detection is critical and thus, the off-line approach is inapplicable due to the processes on entire log data. Other proposed approaches that involve the use of specific anomalies limit anomaly detect from unknown types [31]. Concurrent generation of log data poses another challenge, especially for workflow-based approaches where a model generated from a standard workflow can only apply to log messages produced sequentially [24, 11]. Apart from that, most approaches fail to leverage comprehensive log messages including the log key, the timestamp and metric values. Especially, such results in limitation in varied anomaly detection [25, 16, 24, 29].

In this paper, we apply and extend methods from various domain to mitigate and balance the limitations mentioned above. Specifically, we utilize Long Short-Term Memory (LSTM) recurrent neural networks (RNNs) to learn ordered sequences of network traffic representation of a computer network and then construct a new large-scale computer system log dataset consisting of both synthetic and real-world network traffic failures of various error types. Next, we evaluate the ability of the proposed model to detect malicious activity. The experimental results demonstrate that our proposed algorithms can effectively detect patterns of traffic indicative of malicious activity.

2. Related Work

Machine learning (ML) has been widely studied, partly due to a large amount of future promising in a wide range of real applications [9, 26, 27], especially sequence-to-sequence learning tasks [4, 22]. A large number of application in anomaly detection involving large sets of high-dimensional data were forced to use methods less capable of modeling temporal information. Specifically, several re-

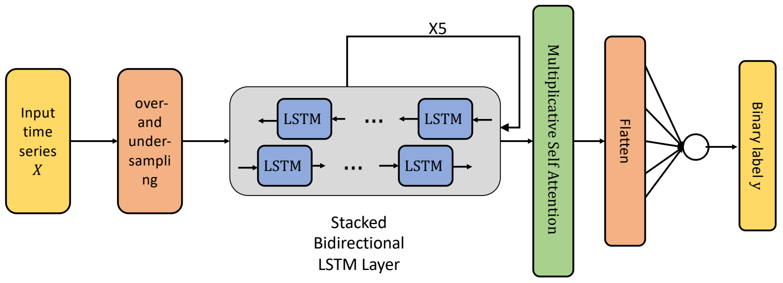
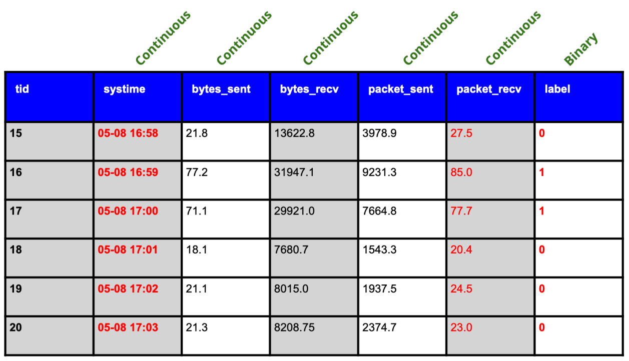


Figure 3. Stacked Bidirectional Self-Attention Network Architecture.

searchers introduced Long Short-Term Memory Networks (LSTM) [6] to perform system log analysis by efficiently processing and prioritizing historical information valuable. Compared with Deep Neural Networks (DNN), LSTMs have shown the superior ability to maintain the memory of long-term dependencies due to a weighted self-loop conditioned on context [28, 17, 12, 33]. It is well-known that the LSTM model can handle high-complexity, temporal, or sequential data in their widespread applications in multiple domains [28, 33].



	Continuous	Continuous	Continuous	Continuous	Continuous	Binary
tid	systime	bytes_sent	bytes_recv	packet_sent	packet_recv	label
15	05-08 16:58	21.8	13622.8	3978.9	27.5	0
16	05-08 16:59	77.2	31947.1	9231.3	85.0	1
17	05-08 17:00	71.1	29921.0	7664.8	77.7	1
18	05-08 17:01	18.1	7680.7	1543.3	20.4	0
19	05-08 17:02	21.1	8015.0	1937.5	24.5	0
20	05-08 17:03	21.3	8208.75	2374.7	23.0	0

Figure 4. The example of filtered data format.

In the anomaly detection context, LSTMs can leverage time-series, non-linear numeric streams of data for anomaly detection. LSTMs could learn a complicated relationship between past data points and current data points. LSTM trained on normal data could capture and model normal behavior of a system and could also handle multivariate time-series data without the need for dimensionality reduction [2]. In addition, LSTM models have proved its ability to model complex nonlinear feature interactions in multivariate time-series data streams by utilizing the shared parameters across time [12, 2].

Zhang *et al.* [31] utilized clustering system logs together with similar format and content to parse streamed console logs and detects early warning signals for failure prediction. Du *et al.* [3] employed customized processing methods on the raw system logs and then to generate language sequences by LSTM networks. Pascanu *et al.* [14] have adopted LSTM networks to preprocess sequences of process application programming interface (API) calls as components to malware detection trained on labeled malware examples.

3. Dataset

We promoted two different datasets to validate anomaly detection algorithms. The datasets we investigate in this study are simulated network traffic dataset and real-life server log dataset.

3.1. Simulated Network Traffic Dataset

We generated simulation of the actual network traffic data to validate the functionality of our methods. We define seven features, termed as *tid*, *systime*, *bytes-sent*, *bytes-recv*, *packet-sent*, *packet-recv*,and *label*, respectively. Data for each time series was assigned a value (0 or 1) and was generated for each minute. Normal data was labeled as 1 for each time series, and then generating random values with normal distribution between 6.25 and 50 around that number. Anomalous data was 10 or 50 standard deviations from the normal noise distribution. Anomalous data deviated from the normal noise distribution by the interval ranging between 2 and 5 standard deviation randomly was generated. Anonymous data was generated throughout the entire time-period and had a maximum duration of 20 hours. An example of generated anomalous data is shown in 2. The simulated network traffic dataset includes around 1090 normal and 400 abnormal data points.

3.2. Real-life Server Log Dataset

The dataset was collected from the remote server operated 7 day. The size of the dataset is around 20GB per day for a single server. Various system logs were saved every 10 seconds. Also, the system summary data was collected on average every 1 second). The processes of events include start, exit, and resource used. In this study, the information

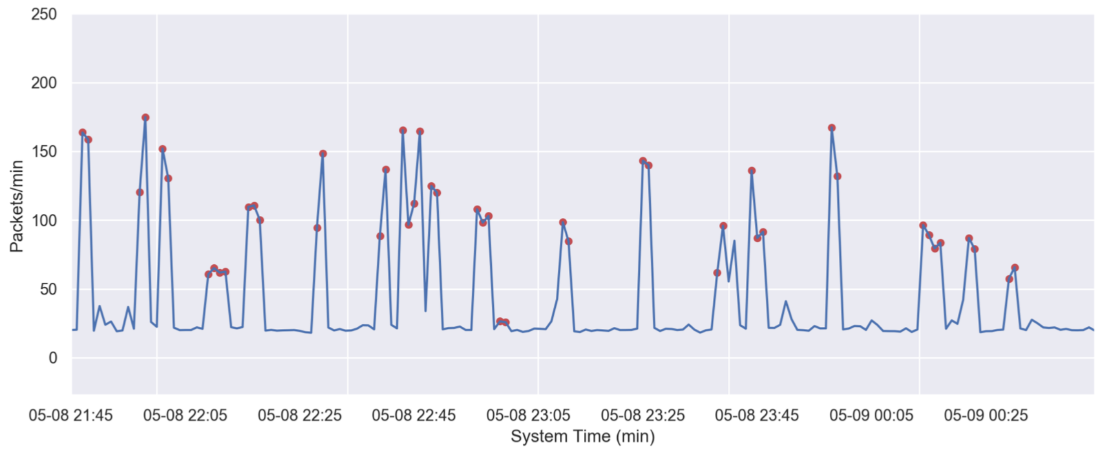


Figure 5. Visualization of examples of simulated data. The abnormal data points were marked by red points.

Table 1. Benchmarking results associated with different approaches over simulated network traffic dataset. Note that the experimental dataset utilizes random injected random errors.

	Pre	*Rec*	F_1 .
LR	0.961	0.890	0.924
DT	0.953	0.988	0.970
SVM	0.974	0.902	0.937
PCA	0.894	0.927	0.910
LSTM	0.998	0.629	0.772
LSTM+Dropout	0.989	0.784	0.875
sLSTM+Dropout	0.929	0.765	0.765

of Network I/O and File I/O, such as cpu, tcp, accept, connect, send, are utilized for data analysis. To generate real-life normal and attack traffic, we configured several hosts, workstations, and servers. The event data is composed of network activities from collected summary data. We define the network in Table. 2. Fig. 2 presents the types of anomalies that we observed in monitoring computer activities. Furthermore, we include more system features such as *tcp-stats*,, *cpu-time-system*,and *cpu-time-irq* to boost the network performance.

4. Methods

In this section, we describe the anomaly detection approach that uses LSTMs to detect anomaly from the time series. Then we discuss the data preprocessing, experimental setup, and evaluation metrics.

4.1. LSTM-AD: LSTM-based Anomaly Detection

Consider a time series $X = \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \ldots, \boldsymbol{x}^{(n)}$, where each step $\boldsymbol{x}^{(t)} \in \mathbb{R}^n$ in the time series is an n-dimensional vector $x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(n)}$, whose elements correspond to input variables [12].

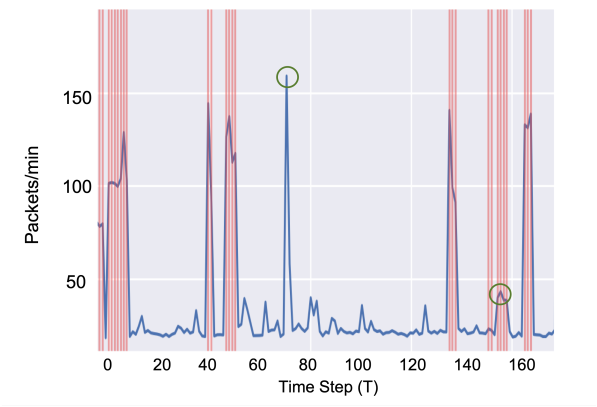
Anomaly detection using the prediction error distribution: With a prediction length of l, each of the selected d dimensions of $\boldsymbol{x}^t \in X$ for $l < t \leq n - l$ is predicted l times. Then we compute an *error vector* $\mathbf{e}^{(\mathbf{t})}$ for point $\mathbf{x}^{(\mathbf{t})}$ as $\boldsymbol{e}^{(t)} = [\boldsymbol{e}_{11}^{(t)}, \ldots, \boldsymbol{e}_{1l}^{(t)}, \ldots, \boldsymbol{e}_{d1}^{(t)}, \ldots, \boldsymbol{e}_{dl}^{(t)}]$, where $e_{ij}^{(t)}$ denotes the difference between $x_i^{(t)}$ and the corresponding value at time $t-j$. Next, we model the error vectors in order to fit a multivariate Gaussian distribution $\mathcal{N} = \mathcal{N}(\mu, \Sigma)$. Then we could formulate the likelihood $p^{(t)}$ of observing an error vector $\boldsymbol{e}^{(t)}$ given by the value of $\mathcal{N}$. The recent work points out that the likelihood is similar to normalized innovations squared (NIS) used for novelty detection using Kalman filter based dynamic prediction model [5]. Also, the error vector with respect to the corresponding points is used to estimate the parameters μ and Σ using Maximum Likelihood Estimation (MLE). An observation $\boldsymbol{x}^{(t)}$ is identified as "normal" if $p^{(t)} > \tau$. On the other hand, the observation is classified as "anomalous". Fig. 1 shows the LSTM model. Then we apply the dropout layer is to prevent the over-fitting to the training dataset. Next, binary cross-entropy is used as a loss function with Adam optimizer. Specifically, the binary cross-entropy loss can be formulated as $loss = -(y \log(p) + (1-y)\log(1-p))$ where y is binary indicator (0 or 1). In this practice, we empirically set $\tau = 0.5$.

4.2. Stacked LSTM based model

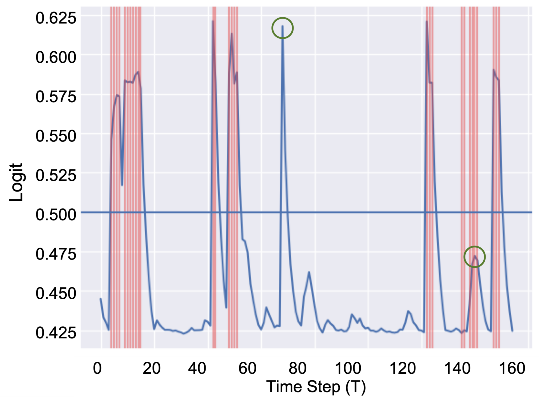
To enhance the detection performance of LSTM model, we stacked 5 LSTM layers, termed as "sLSTM", s.t. since LSTM units are fully connected through recurrent connections, each unit in the lower hidden layer is fully connected

Table 2. Taxonomy of Network attacks and the corresponding characteristics

Attack Type	Characteristics
Denial of service (DoS)	Attempts to block access to system or network resources
General network attack	Maliciously attempt to compromise the security of the network
Password attack	Aims to gain a password by a series of log-in failures
Information gathering attack	Finds known vulnerabilities by scanning or probing computers or networks



(a) Ground Truth



(b) LSTM w/ dropout

Figure 6. Anomaly detection by LSTM model over the simulated network traffic dataset in the certain time interval.

to each unit in the LSTM hidden layer through feed-forward connections.

4.3. Stacked Bidirectional LSTM based model

One shortcoming of conventional LSTMs is that they are only able to leverage previous time state. However, in the anomaly detection tasks, where each program is executed at once, there is no reason not to exploit future correlations as well. Stacked Bidirectional LSTMs (sBiLSTMs) [18] is able to exploit data in both time-sequence directions with two separate hidden layers by making use of past features (via forward states) and future features (via backward states) for a specific time frame [8].

4.4. Attention Mechanism

Given the vector representation of a query q, the attention mechanism uses a parameterized compatibility function $f(\boldsymbol{x}^t, q)$ to measure the dependence between $\boldsymbol{x}^t$ and q^t, or the attention of q to $\boldsymbol{x}^t$ [1]. A softmax function is then applied to transform the alignment score $a \in \mathbb{R}^n$ to a probability distribution $p(z|\boldsymbol{x}, q)$ by normalizing over all the n tokens on q on a specific task, where z is an indicator of which token is crucial to q [8, 20]. That is, a large $p(z = t|\boldsymbol{x}, q)$ means that $\boldsymbol{x}^t$ contributes important information to q. The process can be summarized as follows:

$$a = [f(\boldsymbol{x}^t, q)]_{t=1}^{n}$$
$$p(z|\boldsymbol{x}, q) = softmax(a).$$

Specially,

$$p(z = t|\boldsymbol{x}, q) = \frac{\exp\left(f(\boldsymbol{x}^t, q)\right)}{\sum_{t=1}^{n} \exp\left(f(\boldsymbol{x}^t, q)\right)}$$

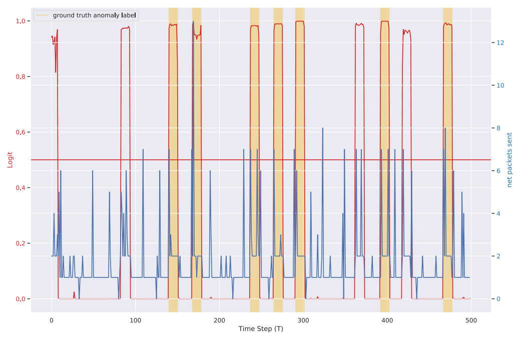
The output s of this attention mechanism is the weighted expectation of a token sampled according to its importance, where the weights are given by $p(z|\boldsymbol{x}, q)$, i.e.,

$$s = \sum_{t=1}^{n} p(z = t|\boldsymbol{x}, q)x^t = \mathbb{E}_{i\sim p(z|\boldsymbol{x},q)(x^t)}$$

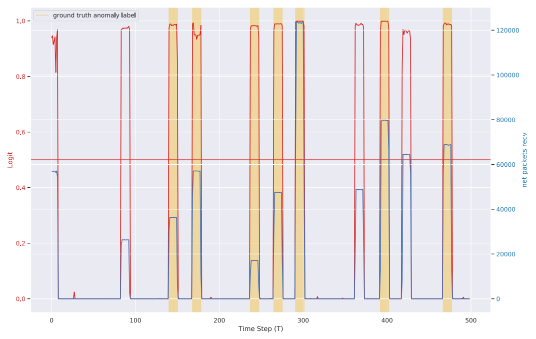
The most commonly used attention mechanisms are additive attention (or multi-layer perceptron attention) [1] and multiplicative attention (or dot-product attention) [23]. They share the same and unified form of attention type, but are different in the compatibility function $f(\boldsymbol{x}^t, q)$. Additive attention is expressed as:

$$f(\boldsymbol{x}^t, q) = W^T \sigma(W^{(1)}\boldsymbol{x}^i + W^{(2)}q)$$

where $W \in \mathbb{R}^n$ and σ are a weight vector and the activation function, respectively.



(a) Packet Sent



(b) Packet Received

Figure 7. Anomaly detection by sBiLSAN model over the real network traffic dataset in certain time interval.

Multiplicative attention, which utilizes inner product or cosine, is expressed as:

$$f(\boldsymbol{x}^t, q) = \langle W^{(1)}\boldsymbol{x}^i, W^{(2)}q \rangle$$

The recent work [19] reports that additive attention often outperforms multiplicative attention in prediction quality. However, the latter is faster and more computational efficient due to optimized matrix multiplication.

4.5. Self-Attention

Self-Attention is an extended version of multi-layer perceptron attention at the feature level. It replaces the vector representation of a query q with a token embedding x_j from the source input itself. It can be applied for both long-range and local dependencies by exploiting latent correlation from elements at different positions for both long-range and local dependencies.

Multi-dimensional attention [23, 7, 19] explores the dependency between $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$ from the same source $\boldsymbol{x}$, and produce context-aware representations. The formula is expressed as:

$$f(\boldsymbol{x}^i, \boldsymbol{x}^j) = W^T\sigma(W^{(1)}\boldsymbol{x}^i + W^{(2)}\boldsymbol{x}^j + b^{(1)}) + b$$

Similar to the p in multi-dimensional attention, each input token $\boldsymbol{x}^j$ is corresponding to a probability matrix p^j, s.t. $p^j_{ki} \triangleq p(z_k = i|\, \boldsymbol{x}, \boldsymbol{x}^j)$. The output representation for $\boldsymbol{x}^j$ is $s_j = \sum_{i=1}^{n} p^j_i \odot \boldsymbol{x}^i$.

4.6. Stacked Bidirectional LSTM based model with Self-Attention

We combine a stacked Bidirectional LSTM network and a self-attention layer to form a Stacked Bidirectional LSTM based model with Self-Attention (sBiLSAN). The network is able to utilize the past input features and future input features in both directions. Also, the self-attention block can produce temporal order encoded and context-aware vector representation for each element/token. The proposed sBiLSAN framework is shown in Fig. 3.

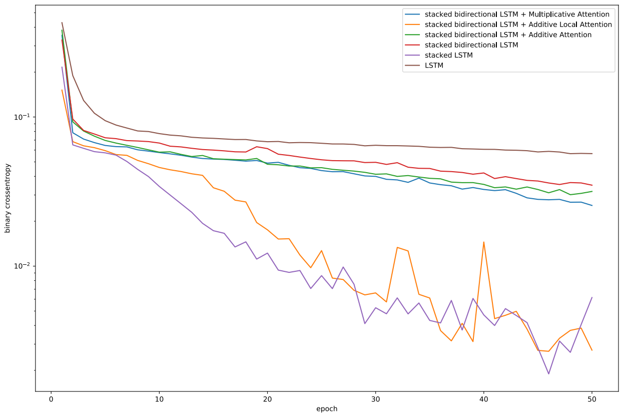


Figure 8. Plots of binary cross-entropy loss versus the number of epochs.

4.7. Data Processing

The system log data is unstructured but contains common event information such as timestamps, and raw traffic (both packet and flows). An unbiased network traffic dataset in a standard format is required. In order to evaluate the anomaly detection system (ADS), an unbiased network traffic dataset in a standard format is required. In general, a packet contains a wide range of raw data, some of which may not be relevant in the context of an ADS. Thus, one of the significant preprocessing steps is to filter irrelevant parameters before capturing and extractions of features from the filtered data. Furthermore, data type conversion, normalization, and discretization are also applied to the anomaly detection mechanism.

Table 3. Benchmarking results associated with different approaches over real network traffic dataset.

	Pre	*Rec*	F_1	*AUC*
LR	0.538	0.689	0.604	0.756
DT	0.473	0.721	0.741	0.741
SVM	0.589	0.541	0.564	0.714
PCA	0.600	0.541	0.569	0.717
LSTM	0.758	0.967	0.850	0.976
sLSTM	0.909	0.962	0.935	0.978
sBiLSTM	0.913	0.960	0.936	0.979
sBiLSTM+Additive Attention	0.792	0.865	0.827	0.927
sBiLSTM+Additive Local Attention	0.847	0.958	0.899	0.975
sBiLSTM+Multiplicative Attention	0.962	0.960	0.961	0.979

In the pre-processing stage, we first sort the events by the timestamps, then group events that were logged in the same second, and create additional features indicating the group information by two nature of attributes, especially *Continuous*, and *Binary*. Intuitively, the logs with timestamps nearby are highly related, information of the nearby logs are also be added to the feature vector. Since the anomalous state happens much less often than the normal state, our data is highly unbalanced. To adjust the class distribution, we used a combination of Synthetic Minority Over-sampling Technique(SMOTE) and Tomek links to clean up overlap between classes on the training data. Note that we observe that applying over-sampling and down-sampling achieve a significantly better result. Please refer to Section 5.2 for more details. Lastly, we visualize the filtered input data format in Fig. 4.

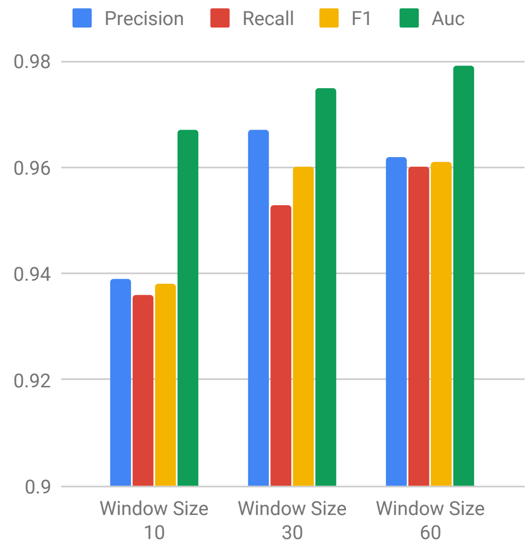


Figure 9. The effects of different history window size on detection accuracy with sBiLSAN.

4.8. Experimental Setup

For the simulated network traffic dataset, we compared nine different *state-of-the-art* methods, including Logistic Regression (LR), Decision Tree (DT), Support Vector Machine (SVM), Principal component analysis (PCA) [24], LSTM [3], LSTM with dropout layer, and stacked LSTM with dropout layer. Next, to further evaluate the robustness of the proposed method in the real scenarios, we conduct the experiments of several popular methods for anomaly detection and prediction. For the anomaly detection, we determine the anomaly of the current timestamp using features from the history data. Also, for predicting network anomaly, we predicted the anomaly of the future status using the feature from the history data. All the models used for comparisons are listed as follows: Logistic Regression (LR), Decision Tree (DT), Support Vector Machine (SVM), Principal component analysis (PCA) [24], LSTM [3], sLSTM, sBiLSTM, sBiLSAN over real network traffic dataset. Note that all parameters of these selected comparing methodologies were set to the suggestions from the original implementation [10]. All experimental codes are implemented in Python with Tensorflow and run on two Nvidia K80 graphics cards.

4.9. Evaluation Metrics

We compared the proposed algorithms with the state-of-the-art over the system information in servers, such CPU, memory, disk I/O, network I/O activities. Furthermore, we utilize the following metrics, namely Precision (*Pre*), Recall (*Rec*), and F_1 score to evaluate the anomaly detection performance of each method in this study. Note that TP, FP, and FN stand for true positive, false positive, and false negatives, respectively.

The formula of Precision (*Pre*) is defined as:

$$Pre = \frac{\mathcal{TP}}{\mathcal{TP} + \mathcal{FP}}$$

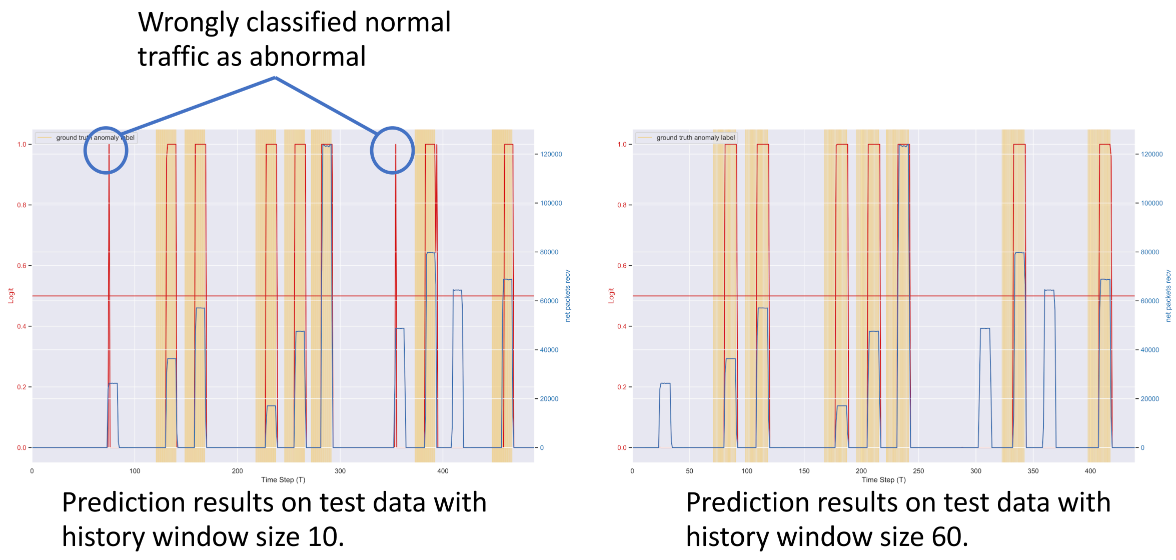


Figure 10. Prediction Visualization Results. Note that the blue line shows the real network activities. The yellow regions are real anomalies marking that the network attacks happened. The red line indicates the output logits from the sBiLSAN model. Data which is greater than the above a threshold (0.5) is classified as anomalies.

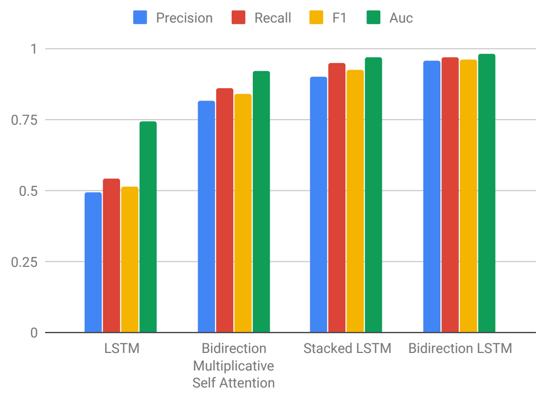


Figure 11. LSTM Prediction Result. Using the features of the previous 60 seconds to predict the state of the future 10 seconds.

The formula of Recall (*Rec*) is defined as:

$$Rec = \frac{\mathcal{TP}}{\mathcal{TP} + \mathcal{FN}}$$

The formula of F_1 score is defined as:

$$F_1 = 2 \times \frac{Pre \times Rec}{Pre + Rec}$$

To better show the performance of a classification model at all classification thresholds, we introduce Area under the ROC Curve (AUC) to measure the entire two-dimensional area underneath the entire receiver operating characteristic curve (ROC).

The simplified formula of AUC score is defined as:

$$\mathrm{AUC} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{1}_{p_i > p_j}$$

Where i denotes all m data points with true label 1, and j denotes all n data points with true label 0. Also, p_i and p_j devote the probability score assigned by the classifier to data point i and j, respectively. $\mathbf{1}$ denotes the indicator function. Specifically, it outputs 1 iff the condition is satisfied (here $p_i > p_j$).

5. Experimental Results

We evaluate and compare the performance of enhanced LSTM model on the simulated and real network traffic datasets to perform anomaly detection and prediction tasks. We use a hybrid approach for anomaly detection and prediction tasks, wherein the neural networks estimate hidden Markov model (HMM) state posteriors.

5.1. Simulated Network Traffic Dataset

We present the results of *the state-of-the-art* methods on simulated network traffic dataset. We report precision, recall, and F_1-score in Table 1. We observe that the conventional LSTM model with a single layer performs very well for the anomaly detection task. With two layers of LSTM model, the performance does not have improvements but have alleviated. The possible reason is that the size of this dataset is small; therefore, LSTM model with a single layer and a large number of memory cells tends to overfit the training data. In the Fig. 6, the normal data point in ground-truth marked by green circle was classified as the abnormal data point by LSTM w/ dropout. Note that the experimental data uses manually injected random errors.

Table 4. The effect of feature size on the vanilla LSTM method over real network traffic dataset. Note feature denotes as FEAT for abbreviation

	Pre	Rec	F_1	$AUC.$
FEAT-4-win-1	0.589	0.989	0.739	0.978
FEAT-4-win-15	0.594	0.988	0.745	0.978
FEAT-4-win-30	0.615	0.984	0.757	0.977
FEAT-4-win-60	0.684	0.936	0.778	0.976
FEAT-33-win-1	0.699	0.957	0.808	0.968
FEAT-33-win-15	0.696	0.954	0.805	0.967
FEAT-33-win-30	0.761	0.955	0.847	0.970
FEAT-33-win-60	0.758	0.967	0.850	0.976

5.2. Real-life Server Log Dataset

Anomaly Detection: From Table 3, in term of F_1 score, LSTM-based methods achieved 40% better than conventional approaches such as decision tree. Also, incorporating more history data, and weakly correlated features could improve the model performance. As shown in Table. 4, when only using network related data for training and testing, the model cannot distinguish the abnormal network traffic due to the network attack with the normal network traffic that sends and gets large files, introducing the false positive in the result. By expanding the training features with the weakly correlated features of net statistics and CPU usage, the model can correctly classify the normal network traffic with high network package bytes received and sent. Overall, adding more features boasts the F1 score for 7% on the vanilla LSTM model. In the Fig. 7, we shows the example of anomaly detection by sBiLSAN model over the real network traffic dataset in certain time interval. Note that the the blue line shows the real network activities. The yellow regions are real anomalies marking that the network attacks happened. The red line indicates the output logits from the sBiLSAN model. Data which is greater than the above a threshold (0.5) is classified as anomalies.

The proposed sBiLSAN produces the best performance in term of F_1 score, which outperforms other *state-of-art* algorithms. The self-attention layer in sBiLSAN implies the existence of the long-range dependency of the history data. Specifically, the self-attention layer can reveal the duration of the current state, which is especially helpful to detect the current state and find anomaly. In Fig. 9, we report the effects of different window size on network performance over the real network traffic dataset. For a fair comparison, we utilize the sBiLSAN network structure as the baseline. It can be observed that the result improves as we increase the window size. It indicates that more feature information can further improve model performance. To visualize the convergence of each model, we calculated the binary cross-entropy loss over the training process for validation. Fig. 8 shows the binary cross-entropy loss versus the number of epochs for the six models.

Anomaly Prediction: We visualize some quantitative results in Fig. 11. The stacked Bidirectional LSTM without self-attention network outperforms that with the self-attention network, which suggests that features extracted from 60 seconds may not provide enough information to predict future states of 10 seconds. It is noted that the stacked Bidirectional LSTM network is robust in anomaly detection and prediction tasks. As shown in Fig. 10, the ground truth anomaly traffic, and normal events are labeled as yellow and red, respectively. Also, the blue peaks uncovered by the yellow region are the normal traffic since they can be differentiated from abnormal traffic with respect to CPU usage and tcp info. It can clearly be observed that in term of the window size of 10, some points of "normal" behavior are mistakenly predicted as abnormal. When the window size increase, all of the future net traffic states can be correctly classified.

6. Conclusion

In this work, we create a real-world network traffic dataset with detailed annotations. Also, we propose a stacked Bidirection self-attention LSTM network (sBiLSAN) for network anomaly detection and prediction tasks. The proposed model learns and encodes entire log messages such as timestamp, tcp stats, and packet values. Besides, we investigate how to incorporate attention mechanism to improve the network performance further. The experiment results demonstrate that sBiLSAN can achieve *state-of-the-art* network performance and outperform existing works (LSTM, etc.) on a wide range of anomaly detection and prediction tasks.

Future work includes exploring more novel applications of the proposed method, such as multi-domains in computer systems. Also, we will investigate how to incorporate a robust algorithm to further improve the performance of anomaly detection and prediction on a large-scale testing set.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] L. Bontemps, J. McDermott, N.-A. Le-Khac, et al. Collective anomaly detection based on long short-term memory recurrent neural networks. In *International Conference on Future*

Data and Security Engineering, pages 141–152. Springer, 2016.

[3] M. Du, F. Li, G. Zheng, and V. Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1285–1298. ACM, 2017.

[4] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[5] P. Hayton, S. Utete, D. King, S. King, P. Anuzis, and L. Tarassenko. Static and dynamic novelty detection methods for jet engine health monitoring. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1851):493–514, 2006.

[6] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] M. Hu, Y. Peng, Z. Huang, X. Qiu, F. Wei, and M. Zhou. Reinforced mnemonic reader for machine reading comprehension. *arXiv preprint arXiv:1705.02798*, 2017.

[8] Z. Huang, W. Xu, and K. Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[9] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[10] Logpai. A log analysis toolkit for automated anomaly detection. `https://github.com/logpai/loglizer`, 2016.

[11] J.-G. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li. Mining invariants from console logs for system problem detection.

[12] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal. Long short term memory networks for anomaly detection in time series. In *Proceedings*, page 89. Presses universitaires de Louvain, 2015.

[13] M. Nicolau, J. McDermott, et al. Learning neural representations for network anomaly detection. *IEEE transactions on cybernetics*, (99):1–14, 2018.

[14] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas. Malware classification with recurrent networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1916–1920. IEEE, 2015.

[15] B. J. Radford, L. M. Apolonio, A. J. Trias, and J. A. Simpson. Network traffic anomaly detection using recurrent neural networks. *arXiv preprint arXiv:1803.10769*, 2018.

[16] S. Roy, A. C. König, I. Dvorkin, and M. Kumar. Perfaugur: Robust diagnostics for performance anomalies in cloud services. In *2015 IEEE 31st International Conference on Data Engineering*, pages 1167–1178. IEEE, 2015.

[17] H. Sak, A. Senior, and F. Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014.

[18] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[19] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[20] T. Shen, T. Zhou, G. Long, J. Jiang, and C. Zhang. Bidirectional block self-attention for fast and memory-efficient sequence modeling. In *International Conference on Learning Representations (ICLR)*, 2018.

[21] J. A. Stankovic. Research directions for the internet of things. *IEEE Internet of Things Journal*, 1(1):3–9, 2014.

[22] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[24] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan. Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 117–132. ACM, 2009.

[25] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda. Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In *Proceedings of the 29th Annual Computer Security Applications Conference*, pages 199–208. ACM, 2013.

[26] C. You, Q. Yang, L. Gjesteby, G. Li, S. Ju, Z. Zhang, Z. Zhao, Y. Zhang, W. Cong, G. Wang, et al. Structurally-sensitive multi-scale deep neural network for low-dose ct denoising. *IEEE Access*, 6:41839–41855, 2018.

[27] C. You, Y. Zhang, X. Zhang, G. Li, S. Ju, Z. Zhao, Z. Zhang, W. Cong, P. K. Saha, and G. Wang. Ct super-resolution gan constrained by the identical, residual, and cycle learning ensemble (gan-circle). *arXiv preprint arXiv:1808.04256*, 2018.

[28] T. Young, D. Hazarika, S. Poria, and E. Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[29] X. Yu, P. Joshi, J. Xu, G. Jin, H. Zhang, and G. Jiang. Cloudseer: Workflow monitoring of cloud infrastructures via interleaved logs. In *ACM SIGPLAN Notices*, volume 51, pages 489–502. ACM, 2016.

[30] D. Yuan, H. Mai, W. Xiong, L. Tan, Y. Zhou, and S. Pasupathy. Sherlog: error diagnosis by connecting clues from runtime logs. In *ACM SIGARCH computer architecture news*, volume 38, pages 143–154. ACM, 2010.

[31] K. Zhang, J. Xu, M. R. Min, G. Jiang, K. Pelechrinis, and H. Zhang. Automated it system failure prediction: A deep learning approach. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 1291–1300. IEEE, 2016.

[32] X. Zhao, K. Rodrigues, Y. Luo, D. Yuan, and M. Stumm. Non-intrusive performance profiling for entire software stacks based on the flow reconstruction principle. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 603–618, 2016.

[33] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu. Text classification improved by integrating bidirectional