ADA: Adaptive Deep Log Anomaly Detector

Yali Yuan*†, Sripriya Srikant Adhatarao*†, Mingkai Lin‡, Yachao Yuan†, Zheli Liu§, and Xiaoming Fu†
†University of Göttingen, Germany. Email:{yali.yuan, adhatarao, fu}@cs.uni-goettingen.de, yachao.yuan@uni-goettingen.de
‡Nanjing University, Nanjing, China. Email: mingkai@smail.nju.edu.cn
§Nankai University, Tianjin, China. Email: liuzheli@nankai.edu.cn

***Abstract*—Large private and government networks are often subjected to attacks like data extrusion and service disruption. Existing anomaly detection systems use offline supervised learning and employ experts for labeling. Hence they cannot detect anomalies in real-time. Even though unsupervised algorithms are increasingly used nowadays, they cannot readily adapt to newer threats. Moreover, many such systems also suffer from high cost of storage and require extensive computational resources. In this paper, we propose ADA: Adaptive Deep Log Anomaly Detector, an unsupervised online deep neural network framework that leverages LSTM networks and regularly adapts to newer log patterns to ensure accurate anomaly detection. In ADA, an adaptive model selection strategy is designed to choose pareto-optimal configurations and thereby utilize resources efficiently. Further, a dynamic threshold algorithm is proposed to dictate the optimal threshold based on recently detected events to improve the detection accuracy. We also use the predictions to guide storage of abnormal data and effectively reduce the overall storage cost. We compare ADA with state-of-the-art approaches through leveraging the Los Alamos National Laboratory cyber security dataset and show that ADA accurately detects anomalies with high F1-score ~95% and it is 97 times faster than existing approaches and incurs very low storage cost.**

***Index Terms*—Anomaly detection, deep neural networks, logs, online training, unsupervised, log-normal, threshold**

I. Introduction

To ensure the security of any online application, it is essential for organizations to detect and mitigate malicious activities on their networks such as, unauthorized access, malwares, port scanning, etc. These attacks may allow unauthorized access to the network and inflict further damages like compromising credentials, violating intellectual property rights, etc. Such attacks may even expose business sensitive information including confidential documents of government agencies, resulting in serious security breaches [1]. The reported losses [2], [3] incurred due to security breach in 2011 by RSA and Target Corporation in 2013 was $66 and $248 millions. Therefore, system logs are generally used to periodically record states of the systems and any significant events at various significant points. Nearly all computer systems today collect and maintain such system-wide log data. These logs are used by experts to diagnose any suspicious behaviours like system failures and unauthorized authentication events. Analyzing system logs is essential to discover the root cause of any problem and potential security breaches. Hence system logs are a valuable source of information and they are crucial for monitoring online applications and detecting anomalies.

Log data analysis and storage are both essential but expensive operations especially in large scale networks. Due to the rapidly increasing number of internet applications and users, the size of log data collected by a system running on even a medium-sized network can grow beyond terabytes per day. Until recently, about 500 MB of logs per day was considered a normal volume in small businesses; today, 5 GB per day is not unusual for such environments. Large organizations can easily produce logs that is orders of magnitude larger than this. With this much information being generated, there is a need for an efficient strategy to store, analyze and manage the system logs. Studies in [4] show that an application running on 51,000 Amazon EC2 instances and publishes five custom metrics will incur a charge of $31,646.40 per month [5]. Therefore, it is challenging and nearly impossible to do manual analysis in real-time and traditional approaches such as mining are proven to be in-effective. Besides, since log data mainly follows a time-series distribution, it is subject to rapid updates. Therefore, obtaining labeled log data for any applications area of interest is often difficult and it is mostly unbalanced or system specific and hence it needs to be pre-processed before analysis.

In addition, obtaining a large-scale log anomaly dataset with high-quality ground truth has been an ongoing challenge [6]–[8]. Labelling log anomalies in a dataset requires experts assistance and therefore it is labor intensive and often expensive. Hence, supervised machine learning strategies like [6]–[8] that depend on prior patterns of normal and abnormal behaviours are not suitable for real-time anaomaly detection systems. Many recent works like [1], [9], proposed unsupervised machine learning algorithms for detecting anomalies. However, these approaches train the models offline and hence cannot effectively adapt to the rapidly changing network behaviours over time and learn new threats and vulnerabilities. Hence, models trained offline can soon become outdated and put the system at risk. Recently, Du et al. [4] proposed online log anomaly detection to generate sequences leveraging Long Short-Term Memory (LSTM) [10] or clustering algorithms for detecting Denial of Service attacks. Some others [11]–[13] leveraged LSTM networks to pre-process the sequence of API calls as components in order to detect malwares in systems. However, these approaches also suffer from increased latency and pre-processing overhead and thus are not suitable for detecting anomalies rapidly.

Therefore, in this paper we propose *ADA: Adaptive Deep Log Anomaly Detector*, an efficient unsupervised online learning approach for detecting anomalies in system logs. Unlike recent works, we exploit the online deep learning [14] to build deep neural networks on the fly with LSTM using unlabelled log data collected from Los Alamos National Laboratory

*Both authors have equal contribution.

(LANL) Cyber Security Dataset [15]. Further, we also propose an adaptive prediction strategy where the pareto-optimal neural network configurations are selected for the current model from the set of all models obtained by online deep learning. In addition, since the patterns in log data are unstable, we develop a dynamic threshold algorithm which uses the recent predictions from the models to dynamically adapt the threshold for detecting abnormal events using the log-normal distribution. Utilizing adaptive strategy and dynamic threshold, ADA always selects the pareto-optimal neural network model with minimal configurations and current threshold based on system environment to predict every event in the system logs. The pareto-optimal policy first learns from the predictions obtained for earlier events. Then, it selects a model that optimizes the needed computational resources. The predictions also drive the decision for selectively storing the log data.

Since we utilize the online deep learning, any system built using the ADA framework can effectively synchronize with any newly discovered log-patterns. Moreover, the system will use models with minimal configurations and predict anomalies with highest possible accuracy using optimal thresholds. In this work, we build an anomaly detection system based on the ADA framework and perform extensive evaluations using the LANL dataset. We show that online deep learning produces highly accurate models where we obtain F1-scores in the range of 0.91 - 0.95 with the latency to predict in the range of 14 ms - 37 ms. We further observed that frequency of abnormal events are far less than normal events in the system logs, so we propose to store only abnormal events after prediction in order to optimally utilize the storage and reduce the overall storage cost. To the best of our knowledge, we are first to propose online deep learning for detecting anomalies in log data with pareto-optimal models and dynamic thresholds.

The contributions in this work include:

- A novel unsupervised anomaly detection framework *ADA: Deep Adaptive Anomaly Detector*, which utilizes online deep learning to build highly accurate models on the fly. ADA also incorporates new log patterns instantly in order to improve anomaly detection.
- An adaptive prediction strategy for selecting the pareto-optimal ADA Event Model (ADA-EM) model to optimize the computational resources and improve the latency of anomaly detection.
- A dynamic threshold technique for reexamining and improving the thresholds of neural network models for detecting abnormal events during anomaly detection.
- A proposal to leverage the insight from event predictions to redirect storage decisions in order to reduce the overall cost of storage.
- Extensive evaluations to show accuracy of the models built with online deep learning, effectiveness of adaptive technique and dynamic threshold in reducing the computational resources and latency compared to state-of-the-art approaches.

II. System Design

In this section, we introduce the design goals and rationale for design choices in ADA followed by the framework.

A. Design Goals

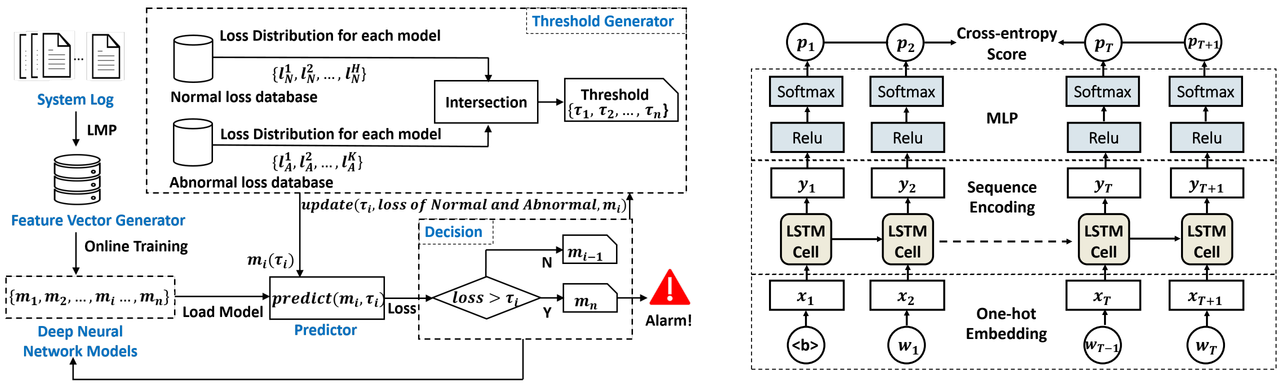
The design of ADA is driven by the following main goals:

1) **Heterogeneous data:** System logs from various applications and systems are heterogeneous in nature and vary significantly in terms of their format and collected information. A generalized anomaly detection system should be able to process and analyze any log format.
2) **Supervision and Data drift:** System behavior and potential threats evolve over time. As a consequence, the system logs and attack models also change. An anomaly detection system should be able to learn newer event patterns automatically even in the absence of experts inputs and labelled dataset.
3) **Accuracy and Threshold:** Anomaly detection systems should correctly predict the abnormal events with high accuracy. In addition, it should adapt system thresholds based on recently observed events and system behaviour to clearly distinguish between normal and abnormal events and reduce false negatives.
4) **Resource utilization:** Since many anomaly detection systems are based on deep neural networks, they demand high computational resources to produce higher accuracy and use parallelization to reduce latency. An efficient anomaly detection system should be able to use modern and established algorithms with minimal resources.
5) **Cost:** System logs are generated everyday and hence the storage cost for these logs also increases with time. An efficient strategy for storage can greatly reduce the overall system cost. Further, many offline algorithms also use increased computational resources to improve accuracy. An anomaly detection system should be able to produce results with good resource-accuracy trade-off.

B. Architectural Components

Fig. 1(a) shows the ADA framework in detail. The components of this framework are described as follows:

- System Logs: This module stores and feeds the logs to the Feature Vector Generator module. The logs are collected from multiple sources in the system and contains information about different system states and events of interest defined by the system administrators.
- Feature Vector Generator: This module uses the Language Model Processing (LMP) [16] to process incoming log streams and generates feature vectors. The resulting features are stored in the feature vector database.
- Models: This module generates online deep neural networks using ADA-EM shown in Fig. 1(b) for detecting anomalies in system logs.
- Predictor: This module initially uses the benchmark model m_i for predicting the incoming log event and sends the prediction results to the decision module. Based on the prediction and adaptive principle shown in



(a) ADA Framework.

(b) ADA Event Model.

Fig. 1: ADA Architecture.

Algorithm 1 (see §II-E), the corresponding next model will be loaded and used for predicting the next event.

– Decision: This module uses the prediction from the predict module and decides whether the predicted event is a normal or abnormal event using the Algorithm 1.
– Threshold Generator: This module stores the recent normal event losses $\{l_N^1, l_N^2, \cdots, l_N^H\}$ and abnormal event losses $\{l_A^1, l_A^2, \cdots, l_A^K\}$ for each model m_i. Using the dynamic threshold computation (see §III), we obtain the current threshold for every model.

C. Tokenization

In order to readily use the arbitrary log formats, we treat every line in the log file as a sequence of tokens. In this work, we consider word-level tokenization granularity in log files. For tokenizing the words, we assume that the tokens in every line in the log are delimited by some known characters (e.g., a space, a comma or a period). We split every ine in the log file based on the delimiter and define a shared vocabulary of "words" over all fields in the log file. Essentially, the vocabulary is composed of the most frequently appearing tokens in the system logs. Further, we treat any missing data as a single feature and use the character "?" to represent it.

D. ADA Event Model (ADA-EM)

In order to efficiently implement the ADA framework, we also propose ADA-EM shown in Fig. 1(b) leveraging LSTM networks. The LSTM architecture was first introduced for machine translation in [17] and since then it has been extensively applied in language processing applications. Therefore, we train the LSTM-based ADA-EM to process instances of normal time-series in the system logs. Specifically, ADA-EM takes the embedded sequences of tokens as input and outputs the distribution for the next token.

We begin with one-hot embedding, which produces unique embedding vectors for every token in the vocabulary set. More specifically, suppose we have a line in the log file with T tokens, and we can describe it as $\mathcal{W} = \{w_0, w_1, \cdots, w_T\}$ where w_0 represents the starting flag $\langle b \rangle$ followed by T tokens of the log line. The layer of one-hot embedding processes the time-series log-line input with $T+1$ tokens $\mathcal{W}$ into one-hot vectors $\mathcal{X} = \{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_{T+1}\}$.

The one-hot embedding is followed by the sequence encoding. Among such large amount of normal events in logs, there exists potential sequential information for language models. To capture this sequential information, an improved LSTM is used in ADA-EM, which is well suited for this task. When applying LSTM recursively to a line in the event log from left to right, the sequence representation will be enhanced progressively with the information from subsequent tokens in this sequence.

Based on the input one-hot vectors, LSTM produces the summary of the past input sequences through the cell state vector $\mathbf{c}_t$. Given $\mathcal{X}$, $\mathbf{y}_t$ is the hidden state of the LSTM cell at time t, which can help to achieve the desired log event prediction. After T times of recursive updates from Eqs. (1) to (5), the hidden representation at token t namely y_t gives a better representation of each token.

$$\mathbf{y}_t = \mathbf{o}_t \circ tanh(\mathbf{c}_t), \tag{1}$$

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{yc}\mathbf{y}_{t-1} + \mathbf{b}_c), \tag{2}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{yo}\mathbf{y}_{t-1} + \mathbf{b}_o), \tag{3}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}\mathbf{x}_i + \mathbf{W}_{yi}\mathbf{y}_{t-1} + \mathbf{b}_i), \tag{4}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\mathbf{x}_f + \mathbf{W}_{yf}\mathbf{y}_{t-1} + \mathbf{b}_f). \tag{5}$$

After that, a fully connected layer with final output unit of vocabulary size produces the output of the model. Once the hidden representations $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_{T+1}\}$ is obtained, we input these vectors into a Multi-Layer Perception (MLP) [18] and get the final probability distribution of the next token. Please note that we use two layers of MLP to achieve better representation capacity in ADA-EM. We use the ReLU activation function for the first layer in order to avoid over-fitting while the softmax function is used for the second layer to normalize the output of MLP layer and get the target output. Given the weight metrics $\mathbf{W}_r, \mathbf{W}_s$ and bias vectors $\mathbf{b}_r, \mathbf{b}_s$ for these two layers, the target output $\mathbf{p}_t$ can be formulated as follows:

$$\mathbf{p}_t = \text{softmax}(\mathbf{b}_s + \mathbf{W}_s(\text{Relu}(\mathbf{b}_r + \mathbf{W}_r(\mathbf{y}_t)))).$$

We use $\frac{1}{T}\sum_{t=1}^{T} H(\mathbf{x_{t+1}}, \mathbf{p_t})$ as the cross-entropy loss function along with resources in ADA-EM for two important purposes: for obtaining an anomaly score for every log line and as the training objective to update the model weights.

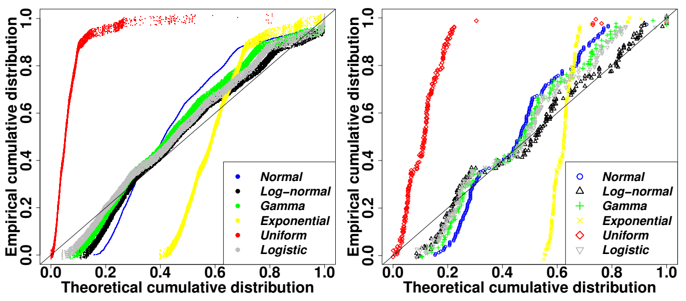
Algorithm 1: Adaptive decision making algorithm

```
Input  : Initialize Pareto-optimal model $m_i$ with its
         configuration $c_i$ and threshold $\tau_i$
Output: Decision, loss
Decision = unknown
for Each log event $d \in D$ do
    loss = Predict($m_i$, $d$)
    if loss $\leq \tau_i$ then
        select $m_{i-1}$
        Decision = Normal
    else
        select $m_n$
        Decision = Abnormal
    end
end
return Decision, loss
```

E. ADA Design

ADA is basically an event driven anomaly detection framework that detects abnormal events in system logs. With an assumption that initial training logs at the first hour represents normal behavior we begin the online training to build deep neural network models similar to Doyen et. al [14]. We start with building a shallow LSTM model with this normal log data. As the time progresses, the number of models also increase with each model having deeper layers than the previous model in time. The goal is to build a deep neural network model from the captured logs in real-time and at the same time use these models to predict any anomalies in the logs. We acknowledge that at the beginning when the model is not trained well, high false positives are observed, however, as we build deeper models, the accuracy gets better over time. Further, models can be added or updated according to the variations observed during predictions. For instance, during the first four hours, we generated a model m_1 with 128 LSTM layers and this model produced a F1-score of ~91% (see §IV). With the increasing number of log samples, the number of layers was also increased and we generated a second model m_2 with 258 layers which produced an F1-score of ~93%. This process is repeated until a deep enough model that produces the desired highest accuracy is obtained. We sort all the generated models based on their depth and represent them as a set $\{m_1, m_2, \cdots, m_n\}$ hereafter.

In ADA, every model in the model set is assigned a corresponding threshold value (see §III) for distinguishing the normal events from abnormal events. During decision making we use the Algorithm 1, and every prediction is measured against the threshold for normal events established for the model. When an abnormal event is detected, the system immediately raises an alarm. The resulting decision drives the next model that should be used for predicting the next event in the log. By virtue of the adaptive principle as described in Algorithm 1, whenever a normal event is found we select a model that is shallower than the current model and a deeper model otherwise. The resulting prediction loss is forwarded to the threshold generator for any required updates.



(a) Normal Distribution. (b) Abnormal Distribution.

Fig. 2: Loss Distribution Fit.

In this work, we define an event as the operational unit of work for any system process which has a finite set of action sequences. Suppose that we have a set of deep neural network models where each model m_i has n_x layers. Each model m_i has one configuration $c_i, i \in n$. $\mathcal{C}$ is the set for all possible configurations of models, $\mathcal{C} = \{c_1, c_2, \cdots, c_n\}$. For each configuration of c_i, we have two interested mappings: a mapping from c_i to its computation resource $R(c_i)$ and to its latency measure $T(c_i)$. ADA searches for Pareto-optimal set $\mathbb{P}$, such that there is no alternative model $m_i^{'}$ with a configuration $c_i^{'}$ that requires less computation resource (R) and offers lower latency (T). Formally, $\mathbb{P}$ is defined as follows:

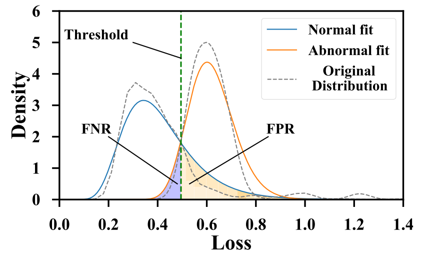
$$\mathbb{P} = \{c_i \in \mathcal{C} : \{c_i^{'} \in \mathcal{C} : R(c_i^{'}) < R(c_i), T(c_i^{'}) < T(c_i)\} = \varnothing\}$$

As mentioned before, we treat log-lines as sequence of tokens and ADA learns normal behavior for a set of users who produced a stream of system logs as follows:

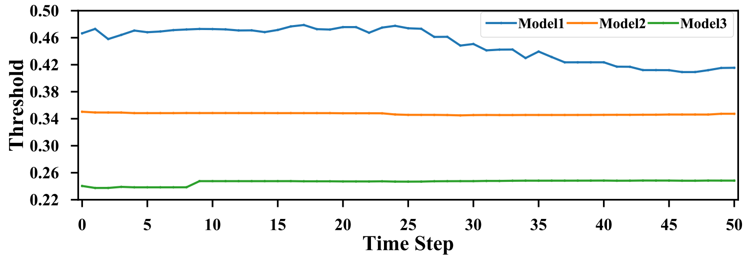
1) When a log stream arrives, the language processing model (LMP) is first utilized to abstract the tokens and build the log feature vector.
2) The above discussed pareto-optimal policy is used to select the optimal model m_i for the current system state.
3) The next model is selected by the feedback of prediction results from the current model following the Algorithm 1. If the prediction result is normal, we select m_{i-1} which is the pareto-optimal model as it is shallower than current model and hence uses less computation and incurs less latency. Otherwise, the model m_n which is the deepest available model is chosen as it provides higher accuracy and is desired when an abnormal event is detected at the cost of increased computation.
4) The threshold τ is selected for each model based on the loss distribution of normal and abnormal data processed by the model and is discussed in the following section.

III. Threshold Computation

The behavioural patterns in system logs change over time as newer threats and attacks are introduced to the system. Thus, predetermined threshold for normal and abnormal behaviours become inadequate and need to be updated regularly. Therefore, in this work we introduce a new dynamic threshold computation technique that automatically learns from the on-



(a) Log normal distribution of log loss data.



(b) Threshold update over time steps.

Fig. 3: ADA Dynamic Threshold.

going behaviours in the system and adjusts the corresponding threshold for detecting anomalies.

Essentially, we inspect the loss distribution of logged events to automatically determine the threshold for every ADA-EM model. During the initial phase, for instance, the first hour, we only have normal log loss distribution and hence in order to compute the initial threshold we use a small number of abnormal events ground truth from the LANL dataset. Subsequently, once the models are introduced to more incoming events, the initial threshold and the loss distribution of normal and abnormal events are updated dynamically to reflect the observed behaviour in the log stream.

We studied several classical loss distributions and employed the Probability-Probability (P-P) plot to assess the appropriate fit for LANL dataset and find the optimal distribution for determining the threshold. In Fig. 2 we plot the empirical distribution of losses obtained from the ADA-EM models against the best fitting theoretical distributions. From this figure, we can clearly see that for both normal and abnormal loss distributions, log normal is the best fit. Therefore, in ADA, we consider that log loss data follows the log normal distribution. In the Fig. 3(a) we show the losses with log-normal distribution for randomly chosen 10000 normal and 200 abnormal log events from the dataset. It is clear that point of intersection of both fits yields the best threshold choice in Fig. 3(a), as the sum of True Positive Rate (TPR) and False Positive Rate (FPR) are minimum (see Proposition 1).

Definition 1: For both normal and abnormal losses obtained by the ADA-EM models, the loss distributions are fitted by log normal distribution. Hence, the probability density function of real log data $p^{loss}(x)$ can be estimated by log normal density function $f^{\mu,\sigma}(x)$:

$$p^{loss}(x) \approx f^{\mu,\sigma}(x),$$

where,

$$f^{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma x} exp(-\frac{(log(x)-\mu)^2}{2\sigma^2}), x > 0.$$

The parameter μ and σ are estimated via maximum likelihood estimation based on the loss dataset. The challenge now is to find an optimal threshold based on the fits of losses. The loss threshold finding problem can be cast as the optimization problem formulated below,

$$\min_x \mathcal{L}(x) = (1 - F_X^N(x)) + F_X^A(x), x > 0, \quad (6)$$

where $F_X^N(x)$ and $F_X^A(x)$ are the Cumulative Distribution Function (CDF) of log-normal distribution [19]. X is the variable and x is loss value of the variable X.

Proposition 1: Two different log-normal densities always have an overlapping section with the condition that $\beta^2 - 4\alpha\gamma \geq 0$, where $\alpha = \frac{1}{2\sigma_2^2} - \frac{1}{2\sigma_1^2}$, $\beta = \frac{\mu_2}{2\sigma_2^2} - \frac{\mu_1}{2\sigma_1^2}$ and $\gamma = \frac{\mu_2}{2\sigma_2^2} - \frac{\mu_1}{2\sigma_1^2} - log(\frac{\sigma_1}{\sigma_2})$. The loss threshold $x \in \{x_1, x_2\}$ that satisfies the Eq. (6) is the optimal loss threshold x^*, where x_1 and x_2 are intersections of both log normal fits $f_N^{\mu_1,\sigma_1}(x)$ and $f_A^{\mu_2,\sigma_2}(x)$.

Proof 1: Let μ_1, σ_1 and μ_2, σ_2, where $\sigma_1, \sigma_2 > 0$, be the corresponding parameters of the density functions. Then

$$\frac{1}{\sqrt{2\pi}\sigma_1 x} exp(-\frac{(log(x)-\mu_1)^2}{2\sigma_1^2}) = \frac{1}{\sqrt{2\pi}\sigma_2 x} exp(-\frac{(log(x)-\mu_2)^2}{2\sigma_2^2}). \quad (7)$$

Applying the logarithm to Eq. (7), we have,

$$log(\frac{\sigma_1}{\sigma_2}) - (-\frac{(log(x)-\mu_2)^2}{2\sigma_2^2}) + (-\frac{(log(x)-\mu_1)^2}{2\sigma_1^2}) = 0. \quad (8)$$

Eq. (8) can be written in terms of a quadratic function in $log(x)$,

$$\alpha(log(x))^2 + \beta log(x) + \gamma = 0, \quad (9)$$

where $\alpha = \frac{1}{2\sigma_2^2} - \frac{1}{2\sigma_1^2}$, $\beta = \frac{\mu_2}{2\sigma_2^2} - \frac{\mu_1}{2\sigma_1^2}$ and $\gamma = \frac{\mu_2}{2\sigma_2^2} - \frac{\mu_1}{2\sigma_1^2} - log(\frac{\sigma_1}{\sigma_2})$.

Now, according to Definition 1, $\mathcal{L}(x)$ can be written in,

$$\begin{aligned} \mathcal{L}(x) &= (1 - F_X^N(x)) + (F_X^A(x)) \\ &= P_N(X \geq x) + P_A(X \leq x) \\ &= \int_x^\infty f_N^{\mu_1,\sigma_1}(x) + \int_0^x f_A^{\mu_2,\sigma_2}(x). \end{aligned} \quad (10)$$

Since $f^{\mu,\sigma}(x)$ is the continuous function in interval $(0, +\infty)$, the first order partial derivative of $\mathcal{L}(x)$ with respect to x, for $x > 0$, is given as,

$$\begin{aligned} \frac{\partial \mathcal{L}(x)}{\partial x} &= -f_N^{\mu_1,\sigma_1}(x) + f_A^{\mu_2,\sigma_2}(x) \\ &= -\frac{1}{\sqrt{2\pi}\sigma_1 x} exp(-\frac{(log(x)-\mu_1)^2}{2\sigma_1^2}) \\ &+ \frac{1}{\sqrt{2\pi}\sigma_2 x} exp(-\frac{(log(x)-\mu_2)^2}{2\sigma_2^2}). \end{aligned}$$

Let $\frac{\partial \mathcal{L}(x)}{\partial x} = 0$, then we get x_1 and x_2 as given in Eq. (9) are candidates loss points. We observe that Eq. (9) can have at most 2 solutions. Furthermore, it can be shown that two different x_1 and x_2 fulfill,

$$x_1 = e^{\frac{-\sqrt{\beta^2-4\alpha\gamma}-\beta}{2\alpha}}, \qquad x_2 = e^{\frac{\sqrt{\beta^2-4\alpha\gamma}-\beta}{2\alpha}},$$

where $\beta^2 - 4\alpha\gamma \geq 0$. The optimal threshold is computed as the intersection at $x = x^*$ that fulfills Eq. (6), which concludes the proof of Proposition 1.

TABLE I: LANL dataset statistics.

Datasets	Events	Source Computer	Destination Computer
Authentication	1,051,430,459	16,230	15,895
Red team	749	4	301

The variations for the proposed dynamic threshold depend on the recently observed log patterns. In this work, during experiments, we record the most recent 10000 normal event losses and 200 abnormal event losses and update the threshold whenever more than 20% of the recorded losses changed. As such, the loss for model m_1 ranges from 0.1 - 3.1 and the threshold ranges from 0.41 - 0.46. Since normal log data occupied the majority in the whole dataset, more than 80% of log event losses are clustered between 0.1 - 0.4. Correspondingly, the loss for model m_2 ranges from 0.1 - 3.0 and the threshold ranges from 0.34 - 0.35. Whereas the loss for model m_3 ranges from 0.1 - 2.0 and the threshold ranges from 0.23 - 0.24. In ADA, as the time increases, threshold for each model is updated to adapt to newly observed log patterns.

In Fig. 3(b) we provide detailed illustration of the thresholds updated for three different models over a span of 50 time steps where each time step is a composition of 200 log events. Since we employ the proposed adaptive strategy from Algorithm 1 during the decision making, the pareto-optimal policy selects the less computationally intensive shallower model m_1 to predict more than 80% of the whole test data. Essentially, we use shallow models for predicting normal data and switch to deeper models only when we detect abnormal data. Therefore m_1's threshold is subjected to more changes than the other models. Subsequently, since the observed abnormal events are small, the deeper model m_3 is rarely used and hence its threshold remains almost stable during the testing.

IV. Evaluation

In this section we evaluate the ADA framework and compare its performance with the state-of-the-art approaches.

A. Dataset

In this work, we use the Los Alamos National Laboratory (LANL) Cyber Security Dataset [15] to evaluate the ADA framework. The LANL dataset consists of Windows-based system authentication event logs from LANL's internal computer network. These logs were collected for a period of 58 consecutive days. The dataset contains over one billion log entries comprising of authentication, network flow, DNS lookup events and processes. Privacy related fields such as users, computers, and processes are anonymized in the dataset.

The network activities recorded in the system logs include both normal operational network activities as well as a series of abnormal activities termed as the *red team* activities. The red team activities mainly represent compromised account credentials over a period of 30 days. Further, we used authentication events from the dataset, and the corresponding statistics are summarized in Table I.

B. Experimental Setup

In ADA, using the online deep learning similar to the algorithm described by Sahoo et al. [14], we build multiple ADA-EM neural network models using LSTM and generate features with LMP [16].

For the evaluation, we build four models in ADA with 128, 256, 512 and 1024 LSTM layers and compare them to the state-of-the-art approaches. In order to demonstrate the ability of the models, we limit the scope of the LANL dataset to events recorded on day 8, as it contains the largest number of abnormal events (261) in over seven million system log events. All experiments are performed on a MacBook Pro laptop, with Intel Core i5 CPU at 2.9 GHz and 8 GB (LPDDR3 2133 MHz) of RAM.

C. Baselines

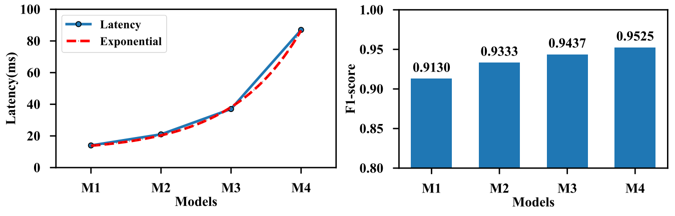
In this work, the proposed ADA framework with the strategies for adaptive model selection and dynamic threshold are employed to build an online unsupervised anomaly detection system. Further, the model m_3 performs optimally in terms of F1-score and latency and hence we select m_3 as the benchmark model in ADA when applying the adaptive algorithm. We compare the performance of this ADA system to the following baselines.

State-of-the-art unsupervised deep learning approaches:

1) NO-ADA: The NO-ADA system is a variant of the ADA framework with online training and unsupervised learning, however, without the adaptive algorithm and dynamic thresholds. The performance of this baseline is obtained with the benchmark model m_3.
2) Fixed-ADA: The Fixed-ADA system is also a variant of the ADA framework with online training and unsupervised learning. In addition, it uses the adaptive strategy shown in Algorithm 1 but the threshold of each model is kept constant like NO-ADA.
3) RNNA [20]: The RNNA system uses the state-of-the-art online and unsupervised learning approach where an anomaly detection system is constructed using LSTM networks with attention algorithms.
4) Kitsune [21]: The Kitsune system uses the state-of-the-art online and unsupervised learning approach that uses the Autoencoder algorithms [22] to differentiate between normal and abnormal events.
5) DAGMM [23]: The DAGMM system uses the state-of-the-art offline unsupervised learning approach called the deep autoencoding gaussian mixture model.

State-of-the-art unsupervised classical learning methods:

1) Isolation Forest (IF) [24]: It is an ensemble-based method for detecting outliers in the dataset.
2) One-class SVM (OCSVM) [25]: It is a max-margin based [26] incremental unsupervised outlier detection model.
3) Self Organizing Maps (SOM) [27]: The SOM networks aim at dividing p-dimensional input space into a finite number of partitions. The whole process is unsupervised



(a) Latency. (b) F1-Score.

Fig. 4: ADA Online Model Performance.

and it is carried out by presenting the vectors to all encompassing neurons.

4) Angle-Based Outlier Detection (ABOD) [28]: ABOD detects the anomalies by the variance of angles between pairs of data samples.

D. Metrics

The following metrics are used to evaluate the performance of ADA and the baseline state-of-the-art approaches.

1) F-measure (F1-score): It is the harmonic mean of precision and recall [29] and is given as follows:
$$\text{F1-score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}},$$
where the precision is defined as,
$$\text{Precision} = \frac{TP}{TP + FP},$$
where TP is True Positives and FP is False Positives.

2) Recall: It is a measure of how many instances were identified correctly and it is given as follows:
$$\text{Recall} = \frac{TP}{TP + FN},$$
where FN is False Negatives.

3) Accuracy: It is a measure of how correctly an anomaly detection system operates by measuring the percentage of TP, True Negatives (TN) along with the number of false alarms in terms of FP and FN that the system produces [30] and is given as follows:
$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}.$$

4) Latency: The Latency is measured from when a log event is sent as input to the model and until the output prediction is obtained.

E. Online Model Performance

To evaluate the performance of ADA, we built four ADA-EM models for the following experiments using online deep learning algorithm similar to sahoo et al. [14]. We begin by training the first model m_1 with 128 LSTM layers followed by the model m_2 with 256 LSTM layers, model m_3 with 512 LSTM layers and the model m_4 with 1024 LSTM layers.

To ensure that models built using online deep learning operated efficiently before applying the adaptive strategy from Algorithm 1 and dynamic threshold discussed in §III we measured the latency and F1-score of the models. For Latency we measured the execution time incurred by the models for predicting an event and the results are shown in Fig. 4(a). Noticeably, model m_1 had the lowest latency of 14ms as it had only 128 LSTM layers while model m_4 with 1024 LSTM layers had the highest latency of 87 ms. We noticed that as number of layers increased, latency increased exponentially.

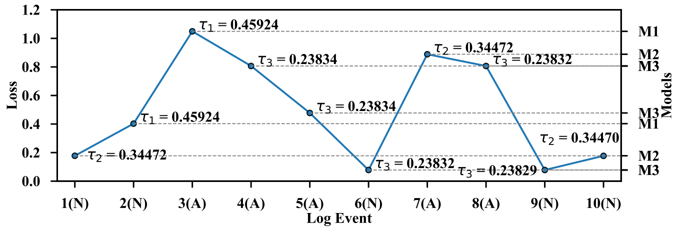


Fig. 5: Adaptive strategy with dynamic Threshold.

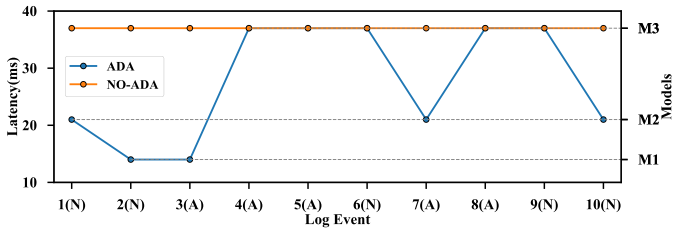


Fig. 6: Latency with ADA vs NO-ADA.

Further, we also measured the F1-score of the models and results are shown in Fig. 4(b). We observed that, as the number of layers increased, the depth of the model also increased and hence the deeper models produced a better F1 score than their shallower counterparts. However, the deeper models have higher latency and demand more computational resources than shallower models. Therefore we only consider models m_1 - m_3 for the evaluation as they had lower latency and produced acceptable F1-scores. Before proceeding further, the proposed adaptive strategy shown in Algorithm 1 needed a benchmark model to begin with. Therefore, we selected the model m_2 as our benchmark model by considering the trade-off between accuracy and latency of the three models.

F. ADA Performance

We evaluated the performance of ADA using these models by applying the adaptive strategy from Algorithm 1 and dynamic threshold from §III. For these experiments, we randomly selected 10 log events and simulated an attack scenario where the events 1-2 show normal authentications and events 3-5 show unauthorized authentications followed by event 6 which is a normal authentication attempt and events 7-8 again perform unauthorized login attempts and finally the events 9-10 display normal authentications. Following the adaptive principle and dynamic threshold, the resulting model used for predicting every event and the corresponding threshold used for distinguishing normal and abnormal events are shown in Fig. 5 and the normal and abnormal events are indicated on the x-axis with the letters *N* and *A* next to events 1-10. As shown in the Fig. 5, during the initial stage, the benchmark model m_2 was selected for predicting the first event. If this log event was normal, then according to the Algorithm 1, the next pareto-optimal model m_1 was selected for predicting the second log event. However, the computationally intensive model m_3 was selected to predict the fourth event to produce high accuracy as the third event was found to be abnormal. The threshold of the model used for prediction is updated and recalculated according to §III. Since the number of log events

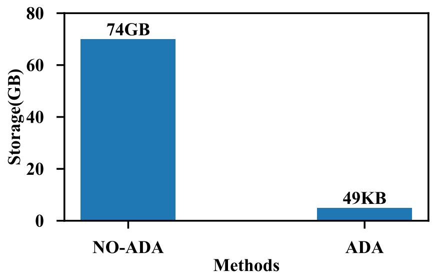


Fig. 7: Storage with ADA vs NO-ADA.

TABLE II: Fixed threshold and adaptive threshold comparison.

Model	F1-score (%)	Accuracy (%)	Recall (%)	Latency (ms)
NO-ADA	94	**94**	**93**	37
Fixed-ADA	91	83	92	18.8
ADA	**95**	91	92	**15.7**

was very small in this experiment, the threshold values did not change that frequently. The results clearly show the benefit of applying the adaptive strategy as the pareto-optimal policy selected shallower models every time a normal event was observed without compromising the identification of abnormal event which were effectively predicted with deeper models to produce higher accuracy.

We also compared the performance of ADA with NO-ADA variant and the results are shown in Fig. 6. We observed from the results that in NO-ADA, there was no adaptive strategy and hence the pareto-optimal policy selected the deepest neural network model m_3 as it produces the highest accuracy at the cost of increased computational resources and latency. Therefore, with NO-ADA, we achieved an accuracy of 94% at the expense of increased computational resources and latency of 37 ms per log event. However, ADA achieved the accuracy of 91% which is close to m_3 but with less computational resources and 57.6% lower latency.

G. Storage

Further, we also analyzed that in system logs, the percentage of abnormal events are very low in comparison to normal events. However, due to the necessity to train the anomaly detection models, organizations usually store the data for prolonged periods. Therefore in this work we propose to store only abnormal data and the most recently observed loss values for the normal data and abnormal data. This is sufficient for ADA to accurately predict the behaviours in the system logs and dynamically compute thresholds based on current behaviours. The resulting storage required with ADA in comparison to existing methods which for simplicity we refer to as NO-ADA is shown in Fig. 7. It is evident that with ADA we incur far less storage cost as we consume just 48 KB of storage for the LANL dataset in comparison to storing the entire 74 GB of the dataset.

For the above mentioned experiments, in Table II we list the F1-score, accuracy, recall and latency of NO-ADA, Fixed-ADA and ADA variants. We observed that NO-ADA has a good performance in terms of F1-score, accuracy and recall, but at the cost of increased latency due to increased model complexity. Noticeably, ADA achieved the highest F1-score of 95% with the lowest computational latency of 15.7 ms among the three variants. Further, the performance of ADA is very close to NO-ADA in terms of accuracy and recall and much better in terms of latency. This is because normal log events occupied a higher percentage in the system log events. In ADA, due to adaptive strategy shown in Algorithm 1, during normal log events the pareto-optimal policy tends to select the model with less computational cost and latency but still produce acceptable accuracy. In ADA, the best model which produces highest accuracy is only employed when abnormal log event appears as these models demand increased computational resources. The adaptive strategy with fixed threshold also decreases the latency in Fixed-ADA variant. In addition, since the threshold in ADA is updated continuously to adapt to newly observed log patterns in online system, ADA can achieve highest F1-score compared to that in NO-ADA and Fixed-ADA. Moreover, in Fixed-ADA, we observed that the number of False Positives increased during testing and therefore we used the deeper models more often than in ADA.

TABLE III: Machine learning results comparison.

Model	F1-score (%)	Accuracy (%)	Recall (%)	Latency (ms)
SOM [27]	48	92	33	**0.03**
IF [24]	47	88.3	84.5	0.17
OCSVM [25]	37	60	**95**	0.35
ABOD [28]	25	**95**	27	0.8
ADA	**95**	91	92	15.7

H. Performance Comparison

Studies by Mirsky et al. [21] show that offline machine learning algorithms perform considerably better than online machine learning algorithms. Since they have access to the entire dataset during the training, offline algorithms perform multiple passes over the data to build a good model that has learned well. However, constructing neural network models using online algorithm is useful when resources, like the training data, computation and/or memory are limited. Since we utilize online algorithms [14] and enhance the anomaly detection with adaptive model selection and dynamic thresholds, we compared the performance of ADA to both state-of-the-art online and offline algorithms.

In Table IV we summarize the experimental results for the state-of-the-art classical unsupervised machine learning algorithms along with ADA for anomaly detection in system logs. As these classical algorithms use offline learning to train and build the models we used 80% of normal events from day 8 in LANL dataset [15] to train and build models for these algorithms. For testing we used 10000 normal events from the remaining 20% logs and 200 abnormal events.

Overall, we observed that classical unsupervised machine learning methods demanded very low computational resources and produced results with very less latency of ~0.8 ms. However, due to unbalanced and complex nature of the logged events, their F1-scores are very low in comparison to ADA even though they were pre-trained with the ground-truth.

Since ADA is based on unsupervised deep learning, we also compared the performance of ADA with other state-of-

TABLE IV: Machine learning results comparison.

Model	F1-score (%)	Accuracy (%)	Recall (%)	Latency (ms)
SOM [9]	48	92	33	**0.03**
IF [10]	47	88.3	84.5	0.17
OCSVM [11]	37	60	**95**	0.35
ABOD [12]	25	**95**	27	0.8
ADA	**95**	91	92	15.7

TABLE V: Deep learning results comparison.

Model	F1-score (%)	Accuracy (%)	Recall (%)	Latency (ms)
RNNA [20]	86	76	78	530
Kitsune [21]	39	64	55	8.6
DAGMM [23]	44	70	47	**0.5**
ADA	**95**	**91**	**92**	15.7

the-art deep learning algorithms and the results are given in Table VI. Overall, the results presented in Table VI clearly show that ADA outperforms RNNA, Kitsune and DAGMM in terms of F1-score, accuracy and recall. Specifically, the F1-score achieved by ADA is about 11% higher compared to RNNA and 144% higher compared to Kitsune and 116% higher compared to DAGMM, respectively. With regards to accuracy, RNNA, Kitsune and DAGMM's accuracy are lower than ADA by nearly 16%, 30% and 23%. Whereas the recall of RNNA, kitsune and DAGMM are nearly 15%, 40% and 49% less than ADA. As for the latency, RNNA performed the worst since it needed 530 ms to predict just one event. This is due to the design of RNNA where they employ user perspective and use a separate LSTM for each user in the dataset. As the number of users increase the overall latency of the system also increases. Although Kitsune and DAGMM had lower latency compared to ADA and RNNA, they did not perform well w.r.t. other metrics. Therefore we conclude that with ADA we efficiently utilize the available resources and produce highly accurate deep neural network models and consume far less resources compared to the state-of-the-art approaches.

V. Related Work

We classify the related works into following categories.

Supervised learning approaches: Supervised learning techniques like [6], [7], [31], [32] have been widely explored for network anomaly detection. In these methods, both normal and abnormal vectors are required to train a binary classifier to detect future anomalies. The main disadvantage of these approaches is that they heavily depend on an experts' experience to label the log data. However, even for an expert it is challenging to distinguish and define anomalies and is often expensive and labor intensive. Another downside of these approaches is that newer anomalies that were not part of the training data may not be detected.

Unsupervised learning approaches: Recently, Du et al. [4] proposed an online log anomaly detector where customized parsing methods are employed on the logs to generate sequences for LSTM or clustering algorithms to detect DoS attacks. However they require pre-define and customized parsing methods for detection. In [11], [20] authors employed RNN and LSTM attention algorithms to improve the performance

TABLE VI: Deep learning results comparison.

Model	F1-score (%)	Accuracy (%)	Recall (%)	Latency (ms)
RNNA [13]	86	76	78	530
Kitsune [14]	39	64	55	8.6
DAGMM [15]	44	70	47	**0.5**
ADA	**95**	**91**	**92**	15.7

of anomaly detection in logs. However, they are defined from a user's perspective, where each user's behaviours are trained on a separate LSTM and hence the number of LSTM networks increase with the increase in users over time. Mirsky et al. [21] presented kitsune framework by using an ensembles of autoencoders for network anomaly detection. However, kitsune adds a separate feature extraction step which requires a basic understanding of the underlying network protocols. In [9], authors used adversarial training of VAE to detect anomalies in Key Performance Monitors of network but, this is not suitable for online systems since the adversarial training is unstable and difficult to converge. Khatuya et al. [33] proposed ADELE, with an aim to select features from system logs in order to create groundwork for a proactive, online failure prediction system. However, this system can perform only short-term failure predictions in the current environment. Moreover, it needs to be trained for at least a month to compute the anomaly score.

Authors from [34] highlight that, Anomaly detection is time sensitive and decisions have to be made in streaming fashion. This enables the system administrators to intervene in an ongoing attack or fix a system performance issue. In this regard, offline learning strategies like [6], [9], [12], [31], that requires to know the prior pattern of normal and abnormal events are not suitable for the new detection systems and also for existing systems trained on outdated events. Unlike the aforementioned research works, our proposed ADA framework builds anomaly detection systems using unsupervised online learning with the adaptive strategy and computes thresholds dynamically to improve the F1-score and accuracy of anomaly detection, reduce the latency and storage of the overall system.

VI. Conclusion and Future Work

In this work we studied the importance of system logs and anomaly detection systems. We studied the limitations of existing anomaly detection approaches such as increased computational overhead, latency, failure to detect new threats, etc. We then presented ADA, an adaptive deep log anomaly detection framework for efficiently detecting anomalies in system logs. To overcome the limitations of existing works, we proposed to build online unsupervised models with adaptive model selection and dynamic thresholds for improving latency, reduce computational overhead and dynamic updates. Using pareto-optimal policy we always select the optimal model with best configurations according to the current system environment. With experiments we demonstrate that ADA improves the performance of the log anomaly detection significantly, and in particular the F1-score and latency compared to state-of-the-art methods along with decreased storage. As part of future work we will explore and incorporate other deep neural network algorithms e.g., variational autoencoders and verify the benefits compared to current design. In addition, we will collect and integrate log data by increasing the scale and scope to different systems and test the robustness of anomaly detection systems built using the ADA framework.

VII. Acknowledgement

This work has been funded by EU H2020 COSAFE project (Grant ID: 824019) and China Scholarship Council (Grant ID: 201706050095). The authors also appreciate Ricardo Zimmer, Trevor Khwam Tabougua and Shahroz Shahjahan for their helpful inputs on the topic of adaptive threshold.

References

[1] A. Bohara, M. A. Noureddine, A. Fawaz, and W. H. Sanders, "An unsupervised multi-detector approach for identifying malicious lateral movement," in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, pp. 224–233, IEEE, 2017.

[2] N. Weiss and R. Miller, "The Target and other financial data breaches: Frequently asked questions," 2015 (last accessed July 20, 2019). https://fas.org/sgp/crs/misc/R43496.pdf.

[3] TrendMicro, "APT myths and challenges," 2012 (last accessed July 20, 2019). http://blog.trendmicro.com/trendlabs-security-intelligence/infographic-apt-myths-and-challenges/.

[4] 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 (CCS)*, pp. 1285–1298, ACM, 2017.

[5] Amazon, "CloudWatch pricing," 2019 (last accessed July 30, 2019). https://aws.amazon.com/cloudwatch/pricing/?nc1=h_ls.

[6] Y. Li, J. Sun, W. Huang, and X. Tian, "Detecting anomaly in large-scale network using mobile crowdsourcing," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 2179–2187, IEEE, 2019.

[7] B. Le Bars and A. Kalogeratos, "A probabilistic framework to node-level anomaly detection in communication networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 2188–2196, IEEE, 2019.

[8] D. Yuan, H. Mai, W. Xiong, L. Tan, Y. Zhou, and S. Pasupathy, "Sherlog: error diagnosis by connecting clues from run-time logs," in *ACM SIGARCH computer architecture news*, vol. 38, pp. 143–154, ACM, 2010.

[9] W. Chen, H. Xu, Z. Li, D. Peiy, J. Chen, H. Qiao, Y. Feng, and Z. Wang, "Unsupervised anomaly detection for intricate kpis via adversarial training of vae," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 1891–1899, IEEE, 2019.

[10] Keras, "LSTM," 2019 (last accessed July 30, 2019). https://keras.io/layers/recurrent/#lstm.

[11] A. R. Tuor, R. Baerwolf, N. Knowles, B. Hutchinson, N. Nichols, and R. Jasper, "Recurrent neural network language models for open vocabulary event-level cyber anomaly detection," in *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[12] N. Zhao, J. Zhu, R. Liu, D. Liu, M. Zhang, and D. Pei, "Label-less: A semi-automatic labelling tool for kpi anomalies," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 1882–1890, IEEE, 2019.

[13] A. R. Tuor, R. Baerwolf, N. Knowles, B. Hutchinson, N. Nichols, and R. Jasper, "Recurrent neural network language models for open vocabulary event-level cyber anomaly detection," in *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[14] D. Sahoo, Q. Pham, J. Lu, and S. C. Hoi, "Online deep learning: learning deep neural networks on the fly," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 2660–2666, AAAI Press, 2018.

[15] A. D. Kent, "Cyber security data sources for dynamic network research," in *Dynamic Networks and Cyber-Security*, pp. 37–65, World Scientific, 2016.

[16] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71, 2018.

[17] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Internal Representations by Error Propagation*. 1988.

[19] W. J. Dixon and F. J. Massey Jr, "Introduction to statistical analysis.," 1951.

[20] A. Brown, A. Tuor, B. Hutchinson, and N. Nichols, "Recurrent neural network attention mechanisms for interpretable system log anomaly detection," in *Proceedings of the First Workshop on Machine Learning for Computing Systems*, p. 1, ACM, 2018.

[21] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv preprint arXiv:1802.09089*, 2018.

[22] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 37–49, 2012.

[23] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," *ICLR*, 2018.

[24] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, p. 3, 2012.

[25] Y. Chen, X. S. Zhou, and T. S. Huang, "One-class svm for learning in image retrieval.," in *ICIP (1)*, pp. 34–37, Citeseer, 2001.

[26] Y. Kong and Y. Fu, "Max-margin action prediction machine," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 9, pp. 1844–1858, 2015.

[27] L. Aguayo and G. A. Barreto, "Novelty detection in time series using self-organizing neural networks: A comprehensive evaluation," *Neural Processing Letters*, vol. 47, no. 2, pp. 717–744, 2018.

[28] H.-P. Kriegel, A. Zimek, *et al.*, "Angle-based outlier detection in high-dimensional data," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 444–452, ACM, 2008.

[29] A. A. Ghorbani, W. Lu, and M. Tavallaee, *Network intrusion detection and prevention: concepts and techniques*, vol. 47. Springer Science & Business Media, 2009.

[30] S. Axelsson, "The base-rate fallacy and the difficulty of intrusion detection," *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 3, pp. 186–205, 2000.

[31] Y. Yuan, G. Kaklamanos, and D. Hogrefe, "A novel semi-supervised adaboost technique for network anomaly detection," in *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 111–114, ACM, 2016.

[32] R. Bitton and A. Shabtai, "A machine learning-based intrusion detection system for securing remote desktop connections to electronic flight bag servers," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[33] S. Khatuya, N. Ganguly, J. Basak, M. Bharde, and B. Mitra, "Adele: Anomaly detection from event log empiricism," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 2114–2122, IEEE, 2018.

[34] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.