UNSUPERVISED ANOMALY DETECTION FOR CONTAINER CLOUD VIA BILSTM-BASED VARIATIONAL AUTO-ENCODER

Yulong Wang[⋆†] *Xingshu Chen*[⋆‡] *Qixu Wang*[⋆] *Run Yang*[†] *Bangzhou Xin*[†]

[⋆] School of Cyber Science and Engineering, Sichuan University, Chengdu, 610065, China
[†] Institute of Computer Application, China Academy of Engineering Physics, Mianyang, 621900, China

ABSTRACT

The appearance of container technology has profoundly changed the development and deployment of multi-tier distributed applications. However, the imperfect system resource isolation features and the kernel-sharing mechanism will introduce significant security risks to the container-based cloud. In this paper, we propose a real-time unsupervised anomaly detection system for monitoring system calls in container cloud via BiLSTM-based variational auto-encoder (VAE). Our proposed BiLSTM-based VAE network leverages the generative characteristics of VAE to learn the robust representations of normal patterns by reconstruction probabilities while being sensitive to long-term dependencies. Our evaluations using real-world datasets show that the BiLSTM-based VAE network achieves excellent detection performance without introducing significant running performance overhead to the container platform.

***Index Terms*—** variational auto-encoder, deep learning , anomaly detection, Linux container, unsupervised learning

1. INTRODUCTION

As an emerging virtualization technology, the Linux container provides a lightweight, flexible, and high-performance operating-system-level virtual run-time environment [1]. Therefore, container technology has been extensively studied with applications in edge computing, Internet of Things (IoT), big data, and so forth.

Nevertheless, security remains an unavoidable factor limiting the sustainability of containers technology [2]. The resource isolation and control mechanisms for containers are mainly provided by the Linux kernel's intrinsic features, which are insufficient to protect co-resident containers' security and privacy. Consequently, substantial efforts on container security have been made so far, aiming to detect the potential risks inside the containers. Some mainstream mechanisms [3–6] have been proposed to detect anomaly behaviors by monitoring the system calls or the multidimensional resource metrics of processes inside the containers. Unfortunately, the existing anomaly detection approaches require a large amount of manually labeled anomalies to learn a predictor by given observations. To solve these issues, some unsupervised learning algorithms [7–9] have been adopted for time-series anomaly detection to focus on normal patterns instead of anomalies. However, such approaches based on discriminative modeling still require a large number of normal observations and suffer from low accuracy due to the failure of unforeseen patterns learning [10].

More recently, the emergence of deep generative modeling for anomaly detection seems to be a better alternative solution. Among these, a representative one is VAE [11], which utilizes neural networks to model the input's underlying probability distribution and detect the underlying cause of anomaly by deriving the reconstruction of the data [12]. Subsequently, there is a rising trend of adopting VAE for time series anomaly detection. A state-of-art anomaly detection algorithm, referred to as *Donut* [13], achieves promising results than other existing unsupervised approaches by using a sliding window to feed time series data into VAE. Since VAE is not a kind of sequential model essentially, *Donut* is not able to handle the long-term dependencies in the input sequences. Hence, [14] tries to combine VAE with a sequential modeling approach by simply using long short-term memory (LSTM) network to connect the encoder and decoder of VAE serially.

In this paper, a robust and real-time unsupervised anomaly detection system is proposed, termed *Pudding*, for container cloud using system call sequences. The *Pudding* integrates data collecting, preprocessing, generative modeling, and anomaly detection modules in a non-intrusive manner. Regarding generative modeling with long-term dependencies, a hybrid neural network with a solid theoretical explanation that deeply combines BiLSTM and VAE structures by the shared gradient is presented to capture long-term-dependent information. Besides, our method is applicable to the environment without a large number of labeled samples and has a strong perception of the unforeseen normal patterns and unknown abnormal behaviors. Extensive experiments using real data from attacking the original Linux container platform show that, compared to other unsupervised anomaly detection methods, *Pudding* significantly improves the detection performance and the robustness against the long-term dependency-related anomalies.

[‡] Corresponding author: Xingshu Chen (chenxsh@scu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant U19A2081, 61872059, 61502085

2. PRELIMINARY AND PROBLEM STATEMENT

2.1. Preliminary: Variational Auto-encoder

The variational auto-encoder is a typical deep Bayesian network that tries to model the underlying relationship between visible input variable x and the latent variable z, forming an autoencoder-like architecture [11]. Since the true posterior distribution of the latent variable z is intractable, the output of *encoder* is the approximate posterior distribution, denoted by $q_\phi(z \mid x)$. It is always assumed to be Gaussian probability distribution $\mathcal{N}\left(\mu_\phi(x), \sigma_\phi^2(x) I\right)$, where ϕ is the parameters of the *encoder*, $\mu_\phi(x)$ and $\sigma_\phi^2(x) I$ are derived by inference network. The *decoder* is another network denoted by $p_\theta(x \mid z)$, where θ is the parameters of the *decoder*. The input of *decoder* is the latent variable z sampled from the defined latent state distribution $q_\phi(z \mid x)$, and the output is the reconstructed $\hat{x}$, sampled from the conditional probability distribution $p_\theta(x \mid z)$. The optimization goal of the VAE network is to make $q_\phi(z \mid x)$ as close to the real posterior probability $p_\theta(z \mid x)$ as possible. Hence the KL divergence is applied between the two distributions in (1), trying to find parameters ϕ and θ to minimize the divergence.

$$\arg\min_{\theta,\phi} D_{KL}\left(q_\phi(z \mid x) \parallel p_\theta(z \mid x)\right) \tag{1}$$

Since Eq. (1) cannot be calculated directly, the variational inference algorithm [11] is used to transform Eq. (1) into maximizing its evidence lower bound Eq. (2).

$$\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(z \mid x)\right] - D_{KL}\left(q_\phi(z \mid x) \| p_\theta(z)\right) \tag{2}$$

2.2. Problem Statement

The system call sequences directly reflect the behavioral characteristics of processes accessing system resources or requesting services from the operating system. Therefore, the semantic information of the system call sequences from the anomalous container is heavily influenced by the malicious behavior, which shows a periodic or intermittent pattern. Accordingly, *Pudding* aims to carve out the behavior profile of the observed containers through a generative network.

Definition 1. ***Sequential Behaviours*** $\mathcal{S} = \{id, s, period\}$ *is defined as collected system call sequences for container* id *during a specific* $period$. $s = \langle s_1, s_2, \ldots, s_L\rangle : \forall k, 1 \leq k \leqslant L$ *and* $0 \leq s_k \leq N$, *where* L *represents the length of the captured sequence,* N *is the number of system call types, and* s_k *is the system call number with a unique integer.*

Problem 1. *Given a Sequential Behaviour* $\mathcal{S}$ *from a container, we aim to construct a deep generative network (i.e., real-time unsupervised anomaly detector)* $\mathcal{G}$ *consisting of an encoder* $\mathcal{E}$ *and a decoder* $\mathcal{D}$. *The latent variable* $\boldsymbol{z}$ *is learned from the visible input* $\boldsymbol{x}$ *by* $\mathcal{E}(\boldsymbol{x})$, *and the reconstructed sequence* $\hat{\boldsymbol{x}}$ *is generated by* $\mathcal{D}(\boldsymbol{z})$. *The anomaly score function* $f : \boldsymbol{loss}(\boldsymbol{x}, \hat{\boldsymbol{x}}) \mapsto \delta$ *can be quantified by the loss between reconstructed sequence and the input sequence.*

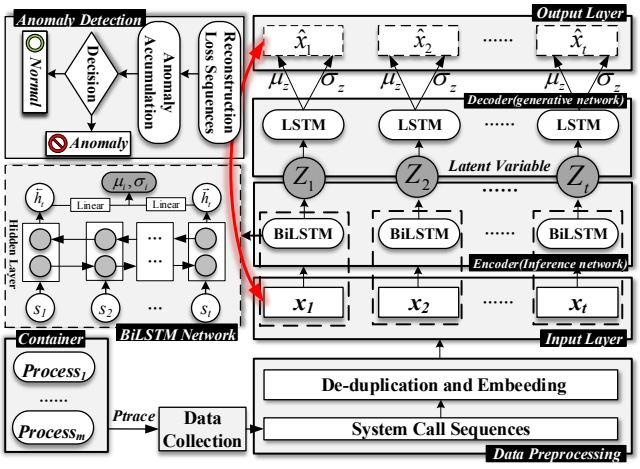

Fig. 1: Overall architecture of proposed system *Pudding*.

3. ANOMALY DETECTION BASED ON BILSTM-VAE NETWORK

3.1. Overall Architecture of *Pudding* System

In order to better understand how proposed system *Pudding* works, we will describe the composition of each component, as illustrated in Fig. 1.

Data Collection. The data collection module is implemented by a Linux system call termed *ptrace*, which automatically senses the creation, operation, and extinction of the container in real-time, then dynamically tracks and collects the lifecycle system call sequences in a non-intrusive manner.

Data Preprocessing. As an efficient word embedding method, word2vec [15] is utilized to map each system call s_k to an associated fixed low-dimensional vector while preserving the contextual information of the sequence. Moreover, in order to reduce the training time overhead, the sequences collected over a long period are de-duplicated.

Generative Network Structure. In the light of aforementioned **problem 1**, the deep generative network $\mathcal{G}$ is constructed by BiLSTM-based variational auto-encoder, where one BiLSTM is utilized as an inferential network (encoder) to estimate the underlying probability distribution of the latent variable z, another BiLSTM is utilized as the generative network (decoder) to sample the reconstructed output $\hat{x}$ from the conditional probability distribution $p(x_t|z_t)$.

Anomaly Detection. The anomaly detector is directly connected to the generation network $\mathcal{G}$ and receives the reconstruction probability served as our anomaly detection scores, which will be applied to make the final decision subsequently.

3.2. Anomaly detection based on BiLSTM-VAE network

3.2.1. BiLSTM-VAE Network Structure

In order to preserve the long-term dependency information, we replace the feed-forward networks in VAE with BiLSTM, which does not require splitting the original sequence to fit the fixed input size of the VAE. Besides, historical states, including past and future, are stored to accommodate multiple time scales and long-term dependencies in BiLSTM networks.

Given the sequential behaviours $\boldsymbol{\mathcal{S}}$, pre-processing is employed to generate the numeric low-dimensional fixed vector representations using word embedding $\boldsymbol{x} = \mathcal{W}(\boldsymbol{\mathcal{S}})$. Regarding the $encoder$ network $\boldsymbol{\mathcal{E}}(\boldsymbol{x}) = q_\phi(z \mid x)$, the latent variable z posterior is chosen to be diagonal Gaussian: $\mathcal{N}\left(\mu_\phi(x), \sigma_\phi^2(x) I\right)$. The forward $\overrightarrow{h}_t$ and backward $\overleftarrow{h}_t$ final hidden states of the LSTM are utilized to estimate $\mu_\phi(x)$ and $\sigma_\phi(x)$, respectively. The means are derived from linear activation, and the standard deviations are derived from softplus activation plus a non-negative number, which are formulated as

$$\mu_\phi(x) = W_\mu^\top \overrightarrow{h}_t + b_\mu \tag{3}$$

$$\sigma_\phi(x) = SoftPlus\left[W_\sigma^\top \overleftarrow{h}_t + b_\sigma\right] + \xi \tag{4}$$

where W and b are the weights and bias in the forward LSTM cell, the $SoftPlus(\cdot)$ activation function denoted by $log\left[exp(\cdot) + 1\right]$ and ξ trick are utilized to prevent that when the local variations are so small, μ_ϕ and σ_ϕ may converge to zero, making the $\log \mu_\phi$ and $\log \sigma_\phi$ unbounded [13]. Besides, $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ are calculated from the forward and backward direction by $\overrightarrow{o}_t \circ \tanh(\overrightarrow{c}_t)$ and $\overleftarrow{o}_t \circ \tanh(\overleftarrow{c}_t)$, where the operator $\circ$ denotes the Hadamard product and o_t is the output gate that controls whether the c_t would be propagated to the h_t.

Regarding the $decoder$ network $\boldsymbol{\mathcal{D}}(\boldsymbol{z}) = p_\theta(x \mid z)$, the calculation process is similar since the network structure is symmetric with the $encoder$ network. Nevertheless, it is worth noting that the stochastic gradient variational Bayes algorithm [11] proposed by Kingma and Welling is adopted, using the re-parameterization trick to sample from the standard normal distribution $\epsilon \sim \mathcal{N}(0, I)$ to rewrite latent variable z as $z(\epsilon) = \mu_\phi(x) + \epsilon \cdot \sigma_\phi^2(x)$ instead of $z \sim \mathcal{N}\left(\mu_\phi(x), \sigma_\phi^2(x) I\right)$. After that, the randomly sampled $z(\epsilon)$ will be fed into the $decoder$ network to estimate $\mu_\theta(z)$ and $\sigma_\theta(z)$ by forward and backward final hidden states, aiming to reconstruct the input $\hat{x}$ as similar as possible to the input x.

According to Eq. (2), the objective of VAE is to maximizing its evidence lower bound. The first term in Eq. (2) is the expectation that can be approximated by Monte Carlo integration [16]. The second term is KL-divergence between two Gaussians, which can be integrated analytically according to the prior distribution of z (*i.e.,* $p_\theta(z)$) is assumed to be $\mathcal{N}(0, I)$. Due to space limitation, we omit the derivation process (refer to [11]) and rewrite the final loss function $\tilde{\mathcal{L}}(\theta, \phi; x)$ of our network:

$$\begin{aligned}\tilde{\mathcal{L}}(\theta, \phi; x) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}\frac{1}{2}\{-1 + \sigma_\phi^{(j)^2}(x_i) + \mu_\phi^{(j)^2}(x_i) \\ - \log \sigma_\phi^{(j)^2}(x_i)\} + \frac{1}{n}\sum_{i=1}^{n}\|x_i - \hat{x}_i\|_\theta\end{aligned} \tag{5}$$

where x_i represents the i_{th} input sequence and n represents the total number of input samples. $\mu_\phi(x_i)$ and $\sigma_\phi(x_i)$ are the output of $encoder$ network, and d denotes the dimensionality of the latent variable. In addition, $\hat{x}_i$ is the reconstructed input generated by the $decoder$ network $p_\theta(x \mid z)$.

3.2.2. Anomaly Detection

In this paper, we utilize the reconstruction term in Eq. (2) as the anomaly score, *i.e.,* $\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(z \mid x)\right]$, defined in [12], which is more principled and objective than reconstruction errors.

More specifically, as shown in Eq. (6), the reconstruction probability can be approximated by Monte Carlo integration by sampling $z^{(m)}$ with M times from $q_\phi(z \mid x)$ for each input sequence x_i and averaging it.

$$\begin{aligned}\mathcal{F}_{score}(x_i) &= E_{z\sim q_\phi(z|x_i)}\left[\log p_\theta(z \mid x)\right] \\ &\approx \frac{1}{M}\sum_{m=1}^{M}\log p_\theta\left(x_i \mid z^{(m)}\right)\end{aligned} \tag{6}$$

Subsequently, the anomaly detection algorithm based on reconstruction probability is given in Algorithm 1. Given the set of the normal dataset X_{trian} after preprocessing, our BiLSTM-VAE network, including $q_\phi(z \mid x) : x \to z$ and $p_\theta(x \mid z) : z \to \hat{x}$, will be trained by optimized the loss function from Eq. (5). Then for each sequence x_{test} to be detected, we first split x_{test} by fixed window size to be compatible with different data collection periods. According to Eq. (6), the reconstruction probability for each sub-sequence $\bar{x}_i$ is approximated by Monte Carlo integration. Finally, the anomaly degree is characterized by the product of the sequence $\mathcal{V}_{rc}$, which is used to determine whether x_{test} is anomaly or not by comparing it to the threshold α.

4. EXPERIMENTS

4.1. Experimental Setup

The UNM public dataset [17] and the system call sequences collected in real container environment are combined to evaluate the proposed method. In addition, we have expanded the actual anomaly events by 1) using sqlmap[1] to attack

[1]The automatic SQL injection and database takeover tool can be found on: `https://sqlmap.org/`.

Algorithm 1 Anomaly detection algorithm based on reconstruction probability

Input: X_{train}, x_{test}, *anomaly threshold* $\boldsymbol{\alpha}$
Output: *Anomaly* or *Normal*

$X_{train}, x_{test} = Preprocessing\left(X_{train}, x_{test}\right)$
$\theta, \phi \leftarrow$ Initialize parameters of BiLSTM-VAE network
$q_\phi\left(z \mid x\right), p_\theta\left(x \mid z\right) \leftarrow train\left(X_{train}, \tilde{\mathcal{L}}\left(\theta, \phi; x\right)\right)$
$\bar{X}_{test} = Splitting\left(x_{test}, window\right)$
Set Reconstruction Probability vector $\mathcal{V}_{rc} = \varnothing$
for $\bar{x}_i$ in $\bar{X}_{test}$ **do**
    $\mu_\phi\left(\bar{x}_i\right), \sigma_\phi\left(\bar{x}_i\right) = q_\phi\left(z_i \mid \bar{x}_i\right)$
    $\mathcal{V}_{rc}\left[i\right] = \mathcal{F}_{score}\left(\bar{x}_i\right)$ ▷ according to Eq. (6)
$\mathcal{T}_{score} = \prod_{i=1}^{|\mathcal{V}_{rc}|} v_i$
if $\mathcal{T}_{score} > \boldsymbol{\alpha}$ **then**
    return x_{test} is normal
else
    return x_{test} is anomaly

the database in the MySQL container, 2) triggering the following kernel vulnerabilities (including CVE-2017-16995, CVE-2017-1000112 and CVE-2017-1000253)[2] to launch container escape attacks. The dataset we built contains a total of 1,315,815 time-series data, of which anomalies account for 0.63%.

4.2. Result Analysis

As shown in Fig. 2, we plot the trend of temporal reconstruction probabilities for normal and abnormal sequences, respectively. It is found that the reconstruction probability of system call sequences generated by normal containers almost appears in the high range. On the contrary, the sequences show a completely different trend while the container is under attack, indicating that our proposed generative modeling is suitable for learning the underlying probability distribution of normal patterns. In addition, the abnormal behaviors are often accompanied by many normal operations; the long-term dependencies of continuous behaviors are fully taken into account by our BiLSTM-VAE network.

To evaluate the detection performance of our method, four representative unsupervised baseline approaches are implemented and compared, including the state-of-art anomaly detection approach [13] in web applications. The experimental results in Table 1 show that our proposed method significantly outperforms all baseline approaches in terms of Accuracy (ACC), Precision (PRE), and F1-measure (F1), while the Recall (REC) is slightly lower. The discriminative approaches like LOF (Local Outlier Factor), isolation forests, and one-class SVM attempt to learn the decision function as a prediction model under a limited number of samples, insensitive to unforeseen patterns. While the VAE-based approach

[2]The Common Vulnerabilities and Exposures (CVE) information in details can be found on: `https://cve.mitre.org/`.

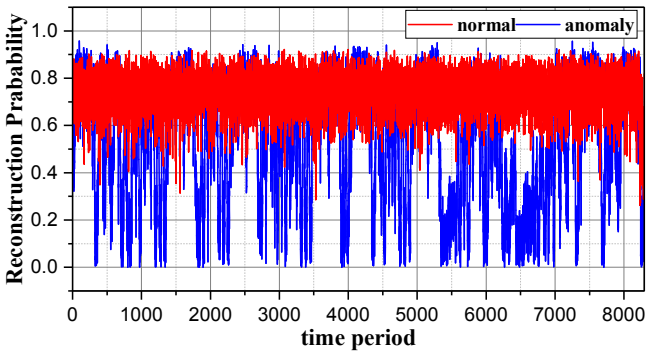

Fig. 2: The trend of temporal reconstruction probabilities for normal and abnormal sequences, respectively.

[13] applies a sliding window to transform the original VAE into a sequential model, the fixed input layer of limited length does not address the issue of long-term dependencies.

Table 1: Detection Performance comparison of different baseline approaches under various evaluation metrics

Approaches	Evaluation Metrics			
	ACC/%	PRE/%	REC/%	F1/%
LOF [18]	79.27	70.89	99.35	82.74
One-class SVM [19]	72.99	69.34	82.23	75.27
Isolation Forests [18]	72.84	64.89	99.52	78.56
VAE-base [13]	77.94	71.32	93.49	80.91
Ours	**90.01**	**84.59**	**97.87**	**90.75**

In addition to detection performance, UnixBench[3] is utilized to compare the run-time performance overhead before and after enabling our *Pudding* system. From the overall perspective, the results indicate our *Pudding* system introduces only 8.41% run-time performance overhead. The benchmarks with a relatively high performance overhead are floating-point operations and the cost of entering and leaving the kernel, which are mainly introduced by *tensorflow* and *ptrace*. Fortunately, the run-time performance will be significantly improved with the optimization of utilized tools.

5. CONCLUSIONS

In this paper, we propose a BiLSTM-based VAE hybrid detection system to detect anomalies for container cloud, named *Pudding*, which can be easily integrated into the container cloud as a security service. Our BiLSTM-based VAE network leverages the generative characteristics of VAE to learn the robust representations of normal patterns by reconstruction probabilities while being sensitive to long-term dependencies. Our evaluations on real-world datasets show that our method achieves excellent detection performance without introducing much performance overhead to the container cloud.

[3]The UnixBench project can be found on GitHub: `https://github.com/kdlucas/byte-unixbench`.

References

[1] Wes Felter, Alexandre Ferreira, Ram Rajamony, and Juan Rubio, "An updated performance comparison of virtual machines and linux containers," in *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2015, pp. 171–172.

[2] Yulong Wang, Qixu Wang, Xingshu Chen, Dajiang Chen, Xiaojie Fang, Mingyong Yin, and Ning Zhang, "Containerguard: A real-time attack detection system in container-based big data platform," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2020.

[3] Rupesh Raj Karn, Prabhakar Kudva, Hai Huang, Sahil Suneja, and Ibrahim M. Elfadel, "Cryptomining detection in container clouds using system calls and explainable machine learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 3, pp. 674–691, 2021.

[4] Zhuping Zou, Yulai Xie, Kai Huang, Gongming Xu, Dan Feng, and Darrell Long, "A docker container anomaly monitoring system based on optimized isolation forest," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2019.

[5] Shujian Ji, Kejiang Ye, and Cheng-Zhong Xu, "Cmonitor: A monitoring and alarming platform for container-based clouds," in *International Conference on Cloud Computing*. Springer, 2019, pp. 324–339.

[6] Amr S. Abed, T. Charles Clancy, and David S. Levy, "Applying bag of system calls for anomalous behavior detection of applications in linux containers," in *2015 IEEE Globecom Workshops (GC Wkshps)*, 2015, pp. 1–5.

[7] Sarah M Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning," *Pattern Recognition*, vol. 58, pp. 121–134, 2016.

[8] Van Loi Cao, Miguel Nicolau, and James McDermott, "One-class classification for anomaly detection with kernel density estimation and genetic programming," in *European Conference on Genetic Programming*. Springer, 2016, pp. 3–18.

[9] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.

[10] Jiahao Bu, Ying Liu, Shenglin Zhang, Weibin Meng, Qitong Liu, Xiaotian Zhu, and Dan Pei, "Rapid deployment of anomaly detection models for large number of emerging kpi streams," in *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2018, pp. 1–8.

[11] Diederik P Kingma and Max Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[12] Jinwon An and Sungzoon Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

[13] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al., "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 187–196.

[14] Shuyu Lin, Ronald Clark, Robert Birke, Sandro Schönborn, Niki Trigoni, and Stephen Roberts, "Anomaly detection for time series using vae-lstm hybrid model," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 4322–4326.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[16] John Geweke, "Bayesian inference in econometric models using monte carlo integration," *Econometrica: Journal of the Econometric Society*, pp. 1317–1339, 1989.

[17] University of New Mexico, "Computer immune systems sequence-based intrusion detection data set," [EB/OL], `https://www.cs.unm.edu/~immsec/systemcalls` Accessed October 05, 2021.

[18] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, pp. 1–39, 2012.

[19] Bernhard Schölkopf, Robert C Williamson, Alexander J Smola, John Shawe-Taylor, John C Platt, et al., "Support vector method for novelty detection.," in *NIPS*. Citeseer, 1999, vol. 12, pp. 582–588.