# Multi-Scale One-Class Recurrent Neural Networks for Discrete Event Sequence Anomaly Detection

Zhiwei Wang[1]*, Zhengzhang Chen[2]†, Jingchao Ni[2], Hui Liu[1], Haifeng Chen[2], Jiliang Tang[1]†
[1]Michigan State University, USA, [2]NEC Laboratories America, USA
{wangzh65,liuhui7,tangjili}@msu.edu,{zchen,jni,haifeng}@nec-labs.com

## ABSTRACT

Discrete event sequences are ubiquitous, such as an ordered event series of process interactions in Information and Communication Technology systems. Recent years have witnessed increasing efforts in detecting anomalies with discrete event sequences. However, it remains an extremely difficult task due to several intrinsic challenges including data imbalance issues, discrete property of the events, and sequential nature of the data. To address these challenges, in this paper, we propose **OC4Seq**, a multi-scale one-class recurrent neural network for detecting anomalies in discrete event sequences. Specifically, **OC4Seq** integrates the anomaly detection objective with recurrent neural networks (RNNs) to embed the discrete event sequences into latent spaces, where anomalies can be easily detected. In addition, given that an anomalous sequence could be caused by either individual events, subsequences of events, or the whole sequence, we design a multi-scale RNN framework to capture different levels of sequential patterns simultaneously. We fully implement and evaluate **OC4Seq** on three real-world system log datasets. The results show that **OC4Seq** consistently outperforms various representative baselines by a large margin. Moreover, through both quantitative and qualitative analysis, the importance of capturing multi-scale sequential patterns for event anomaly detection is verified. To encourage reproducibility, we make the code and data publicly available[1].

## CCS CONCEPTS

• **Computing methodologies** → **Anomaly detection**; *Neural networks*; • **Security and privacy** → Intrusion detection systems.

## KEYWORDS

Anomaly detection; event sequence modeling; one-class recurrent neural network; multi-scale sequential pattern mining

**ACM Reference Format:**
Zhiwei Wang[1]*, Zhengzhang Chen[2]†, Jingchao Ni[2], Hui Liu[1], Haifeng Chen[2], Jiliang Tang[1]†. 2021. Multi-Scale One-Class Recurrent Neural Networks for Discrete Event Sequence Anomaly Detection. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3447548.3467125

## 1 INTRODUCTION

Nowadays, Information and Communication Technology (ICT) has permeated every aspect of our daily life and played a crucial role in society than ever. While ICT systems have brought unprecedented convenience, when in abnormal states caused by malicious attackers, they could also lead to ramifications including severe loss of economy and social wellbeing [12, 24, 33]. Therefore, it is vital to timely and accurately detect abnormal states of ICT systems such that the loss can be mitigated. Fortunately, with the ubiquitous sensors and networks, ICT systems have generated a large amount of monitoring data [5, 11, 12, 14, 24]. Such data contains rich information and provides us with unprecedented opportunities to understand the complex states of ICT systems.

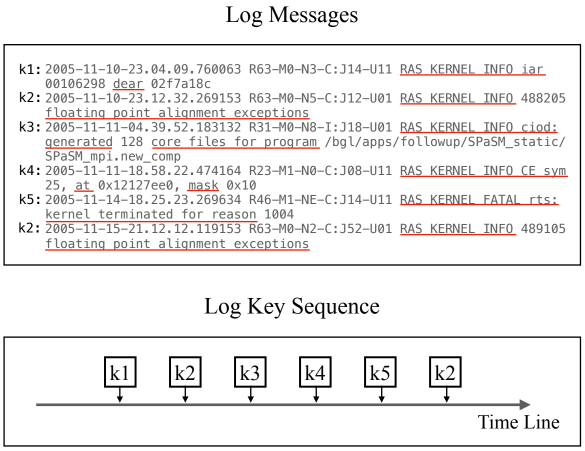
One type of the most important monitoring data is discrete event sequence. A discrete event sequence is defined as an ordered series of events, where each event is a discrete symbol belonging to a finite alphabet [6]. The discrete event sequences can be seen everywhere such as control commands of machine systems, logs of a computer program, and transactions of customer purchases. Due to the rich information they provide, they have been a valuable source for anomaly detection adopted by both academic research and industry practice [4, 6, 11, 12, 14]. For example, system logs that record the detailed messages of run time information of modern computer systems are extensively used for anomaly detection [14, 23, 27, 36]. Each log message can be roughly considered as consisting of a predefined constant print statement (also known as “log key” or “message type”) and a specific parameter (also known as “variable”). When the log keys are arranged chronologically according to the recording time, they form a discrete event sequence that can reflect the underlying system state. Figure 1 illustrates the logs from a BlueGene/L supercomputer system (BGL). In this example, six log messages are generated by five corresponding predefined statements (log keys). These log keys form a discrete event sequence. When the system is in an abnormal state, the resulted discrete event sequences will deviate from the normal patterns. For instance, if “k2” always succeeds “k1” in a normal state, then the log key sequence in Figure 1 may indicate an abnormal state because “k2” is after “k5”, which is unlikely to happen in the normal state. In the following, we refer to discrete event sequences generated by normal and abnormal system states as normal and abnormal sequences, respectively.

Despite the increased interest in detecting anomaly for discrete event sequences [14, 15, 25], the task remains challenging for several reasons. The first challenge is from the data imbalance issue that is commonly observed in anomaly detection related applications. As

[1]https://github.com/wzwtrevor/Multi-Scale-One-Class-Recurrent-Neural-Networks
∗ Work was done during an internship at NEC Laboratories America.
† Zhengzhang Chen and Jiliang Tang are corresponding authors.

*KDD ’21, August 14–18, 2021, Virtual Event, Singapore*

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467125



```
k1: 2005-11-10-23.04.09.760063 R63-M0-N3-C:J14-U11 RAS KERNEL INFO iar
    00106298 dear 02f7a18c
k2: 2005-11-10-23.12.32.269153 R63-M0-N5-C:J12-U01 RAS KERNEL INFO 488205
    floating point alignment exceptions
k3: 2005-11-11-04.39.52.183132 R31-M0-N8-I:J18-U01 RAS KERNEL INFO ciod:
    generated 128 core files for program /bgl/apps/followup/SPaSM_static/
    SPaSM_mpi.new_comp
k4: 2005-11-11-18.58.22.474164 R23-M1-N0-C:J08-U11 RAS KERNEL INFO CE sym
    25, at 0x12127ee0, mask 0x10
k5: 2005-11-14-18.25.23.269634 R46-M1-NE-C:J14-U11 RAS KERNEL FATAL rts:
    kernel terminated for reason 1004
k2: 2005-11-15-21.12.12.119153 R63-M0-N2-C:J52-U01 RAS KERNEL INFO 489105
    floating point alignment exceptions
```

**Figure 1: An illustrative example of BGL log messages and the corresponding log key sequence. Each log message contains a predefined log key that is underscored by red lines.**

systems are in normal states most of the time, abnormal sequences are very rare. This makes the data distribution substantially uneven in terms of normal and abnormal states. Thus, binary classification models that have achieved great success in the other problems become ineffective for anomaly detection. Moreover, in reality, it is hard to obtain prior knowledge about the abnormal sequence, which further exacerbates the situation. The second obstacle comes from the discrete property of events. Unlike continuous sequences where each event is real-valued and has physical meanings, the discrete event sequence consists of meaningless discrete symbols, making it hard to capture the relations of events over time. Finally, the sequential nature of the data makes the problem even more challenging. In order to determine whether a discrete event sequence is abnormal or not, it is essential to consider each event, subsequences of events, and the whole sequence jointly. This requires dedicated efforts to designing models that not only have strong capability to capture the sequential patterns but also are flexible to handle sequential patterns at different scales.

To address the aforementioned challenges, in this paper, we propose **OC4Seq**, a multi-scale one-class recurrent neural network framework for event sequence anomaly detection. It is proposed to directly integrate the anomaly detection objective with a specially designed deep sequence model that explicitly incorporates sequential patterns at different scales. The main contributions of this work are summarized as follows:

- We identify the importance of multi-scale sequential patterns in anomaly detection for discrete event sequences empirically.
- We introduce a novel one-class recurrent neural network framework **OC4Seq** for discrete event sequence anomaly detection. To the best of our knowledge, **OC4Seq** describes the first attempt to incorporate a deep one-class classifier with the event sequence anomaly detection task, and one of the first to extend one-class classifier concepts into applications from data mining fields.
- We propose to directly optimize the deep sequence model with a one-class classification objective. The multi-scale sequence model is trained to explicitly map the local subsequence and whole sequence into different latent spaces, where the normal data points are enclosed by hyperspheres with minimum volume. Our proposed model can be trained in an end-to-end manner.
- The proposed framework is fully implemented and extensively evaluated on three real-world system log datasets. The results show that OC4Seq outperforms the state-of-the-art representative methods with a significant margin.

## 2 PROBLEM STATEMENT

Before we formally define the problem of anomaly detection for discrete event sequences, we first introduce notations that will be used throughout the rest of the paper. Lower-case letters such as $i$ and $j$ are used to denote scalar variables and upper-case letters such as $N$ and $M$ represent scalar constants. Moreover, we use bold lower-case letters to denote vectors such as $\mathbf{v}$ and $\mathbf{x}$ and bold upper-case for matrices such as $\mathbf{W}$. In addition, the $i^{th}$ entry of a vector $\mathbf{v}$ is denoted as $\mathbf{v}(i)$. Similarly, $\mathbf{W}(i,j)$ indicates the entry at $i^{th}$ row and $j^{th}$ column of a matrix $\mathbf{W}$. In the rest of the paper, event and discrete event are used interchangeably. We use $(\cdots)$ to represent an event sequence and subscripts are used to index the events in the sequence such as $(x_1, x_2, x_3)$.

Given an event set $\mathcal{E}$ that contains all possible discrete events, an event sequence $S^i$ is defined as $S^i = (e_1^i, e_2^i, \cdots, e_{N^i}^i)$, where $e_j^i \in \mathcal{E}$ and $N^i$ is the length of sequence $S^i$. Each event $e_j^i$ is represented by a categorical value, *i.e.*, $e_j^i \in \mathcal{N}^+$.

With the notations above, the anomaly detection for discrete event sequence problem under the *one-class setting* is formally defined as follows:

*Given a set of sequences $\mathcal{S} = \{S^1, S^2, \cdots, S^N\}$, where each sequence $S^i$ is normal, we aim to design a one-class classifier that is able to identify whether a new sequence $S$ is normal or not by capturing the underlying multi-scale sequential patterns in $\mathcal{S}$.*

## 3 PRELIMINARIES: ONE-CLASS CLASSIFIER

In this section, we introduce preliminaries that lay a foundation for our proposed framework. A one-class classifier is a specially designed classifier that is trained with objects of a single class and can predict whether an object belongs to this class or not in the test stage. One of the most widely used one-class classifiers is kernel-based such as One-Class Support Vector Machines (OC-SVM) [29] and Support Vector Data Description (SVDD) [31]. Both OC-SVM and SVDD are inspired by SVM that tries to maximize the margin between two classes. Next, we use SVDD as an example to illustrate traditional one-class classifiers. SVDD aims at finding a spherically shaped boundary around the given data in the kernel space. Specifically, let the center of the hypersphere be $\mathbf{c}$ and the radius be $R > 0$. The SVDD objective is defined as follows:

$$\min_{R,\mathbf{c},\varepsilon} R^2 + C\sum_{i=1}^{n} \varepsilon_i \tag{1}$$

$$s.t. \|\phi(\mathbf{x}_i - \mathbf{c})\|^2 \leq R^2 + \varepsilon_i, \varepsilon_i \geq 0, \quad \forall i$$

where $\mathbf{x}_i$ is the feature vector of the $i^{th}$ data and $\varepsilon_i \geq 0$ is a slack variable for $\mathbf{x}_i$ that is introduced to tolerate the presence of outliers in the training set, and the hyperparameter $C$ trades off the errors $\varepsilon_i$ and the volume of the sphere. The objective defined in Equation 1 is a primary form and similar to SVM, it is solved in the dual space by using Lagrange multipliers. For more details of the optimization, please refer to the original paper [31]. Once the $R$ and $\mathbf{c}$ are determined, the points that are outside the sphere will be classified as other classes.

Recently, a deep neural network based one-class classifier called Deep SVDD was introduced in [28]. Inspired by SVDD, it tries to find a minimum hypersphere in the latent space. Unlike SVDD, which relies on kernel functions for feature transformation, Deep SVDD takes advantage of deep neural networks to learn data representations. It employs a quadratic loss for penalizing the distance of every data point representation to the center. The objective function has been proved with nice theoretical properties [28]. Once the neural networks are trained and the center is fixed, outliers will be detected similarly as in SVDD.

## 4 THE PROPOSED FRAMEWORK

In this section, we introduce **OC4Seq**, a multi-scale one-class recurrent neural network framework for event sequence anomaly detection. As illustrated in Figure 2, **OC4Seq** consists of two major components that focus on the global and local information in the sequences, respectively. The details of each component are described in the next subsections.

### 4.1 Learning Embeddings for Events

The inputs of the framework are the sequences of events, where each event $\mathbf{e}_t$ is a one-hot vector and $\mathbf{e}(j) = 1, \mathbf{e}(i) = 0,\ \forall i \neq j$, if $\mathbf{e}_t$ is the $j^{th}$ type event of the set $\mathcal{E}$. In real-world scenarios, event space could be very large, *i.e.*, there are tens of thousands of event types. This can lead $\mathbf{e}_t$ to be very high-dimensional and cause notorious learning issues such as sparsity and curse of dimensionality. In addition, one-hot vector representation makes an implicit assumption that events are independent with each other, which does not hold in most cases. Therefore, we design an embedding layer to embed events into a low-dimension space that can preserve relations between events. To do so, we introduce an embedding matrix $\mathbf{E} \in \mathbb{R}^{d^e \times |\mathcal{E}|}$, where $d^e$ is the dimension of the embedding space and $|\mathcal{E}|$ is the number of event types in $\mathcal{E}$. With the embedding matrix, the representation of $\mathbf{e}_t$ can be obtained as follows:

$$\mathbf{x}_t = \mathbf{E}^T \cdot \mathbf{e}_t \tag{2}$$

where $\mathbf{x}_t \in \mathbb{R}^{d^e}$ is the new low-dimensional dense representation vector for $\mathbf{e}_t$. After the embedding layer, the input sequences will be passed into the other components that will be introduced next.

### 4.2 Anomaly Detection from Global Perspective

To detect an anomalous sequence, it is important to learn an effective representation of the whole sequence in the latent space. To this end, we propose to integrate the widely-used Gated Recurrent Neural Networks (GRU) [7] with one-class objective function. Specifically, given a normal sequence, *i.e.*, $S^i = (\mathbf{x}_1^i, \mathbf{x}_2^i, \cdots, \mathbf{x}_{N^i}^i)$, the GRU learns a representation of the sequence in a recursive manner. At the $t^{th}$ step, the GRU outputs a state vector $\mathbf{h}_t^i$, which is a linear interpolation between previous state $\mathbf{h}_{t-1}^i$ and a candidate state $\tilde{\mathbf{h}}_t^i$. Formally, we have:

$$\mathbf{h}_t^i = z_t^i \odot \mathbf{h}_{t-1}^i + (1 - z_t^i) \odot \tilde{\mathbf{h}}_t^i \tag{3}$$

where $\odot$ is the element-wise multiplication. $z_t$ is the update gate, which is introduced to control how much current state should be updated given the current information $\mathbf{x}_t^i$. $z_t$ is calculated as:

$$z_t^i = \sigma(\mathbf{W}_z \mathbf{x}_t^i + \mathbf{U}_z \mathbf{h}_{t-1}^i) \tag{4}$$

where $\mathbf{W}_z$ and $\mathbf{U}_z$ are the trainable parameters and $\sigma(\cdot)$ is a sigmoid function, which is defined as follows:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

Moreover, the candidate state $\tilde{\mathbf{h}}_\mathbf{t}^i$ introduced in Equation 3 is computed as follows:

$$\tilde{h}_t^i = g(\mathbf{W}\mathbf{x}_t^i + \mathbf{U}(r_t^i \odot \mathbf{h}_{t-1}^i)) \tag{6}$$

where $g(\cdot)$ is the tanh function that is defined:

$$g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{7}$$

$\mathbf{W}$ and $\mathbf{U}$ in Eq. (6) are trainable parameters. $r_t$ is the reset gate. It is introduced to determine how much the candidate state should incorporate previous state. The reset gate is calculated as:

$$r_t^i = \sigma(\mathbf{W}_r \mathbf{x}_t^i + \mathbf{U}_r \mathbf{h}_{t-1}^i) \tag{8}$$

As the state vector $\mathbf{h}_{N^i}$ at the final step summarizes all the information in the previous steps, we regard it as the representation of the whole sequence. Please note that the GRU component can be replaced by any sequence learning models such as Long Short-Term Memory (LSTM) [19]. In fact, we empirically found that the two have similar performance. Due to its structural simplicity, we choose GRU over LSTM. More details of the component analysis can be found in Section 5.

Inspired by the intuition behind the Deep SVDD that all the normal data should be lie within a hypersphere of minimum volume in a latent space, we propose the following objective function to guide the training process:

$$\mathcal{L}_{global} = \min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{h}_{N^i} - \mathbf{c}\|^2 + \lambda \|\Theta\|_F^2 \tag{9}$$

Here, $\mathbf{c}$ is a predefined center in the latent space and $N$ is the total number of sequences in the training set. The first term in the objective function employs a quadratic loss for penalizing the distance of every sequence representation to the center $\mathbf{c}$ and the second term is a regularizer controlled by the hyperparameter $\lambda$. Therefore, this objective will force the GRU model to map sequences to representation vectors that, on average, have the minimum distances to the center $\mathbf{c}$ in the latent space.

Although the global GRU is effective to model the whole sequence, it might ignore vital information for event sequence anomaly detection because of the following reason: the abnormal property of a sequence can be caused by only a small abnormal subsequence or even a single abnormal event. However, when the sequence is long, the abnormal information could be overwhelmed

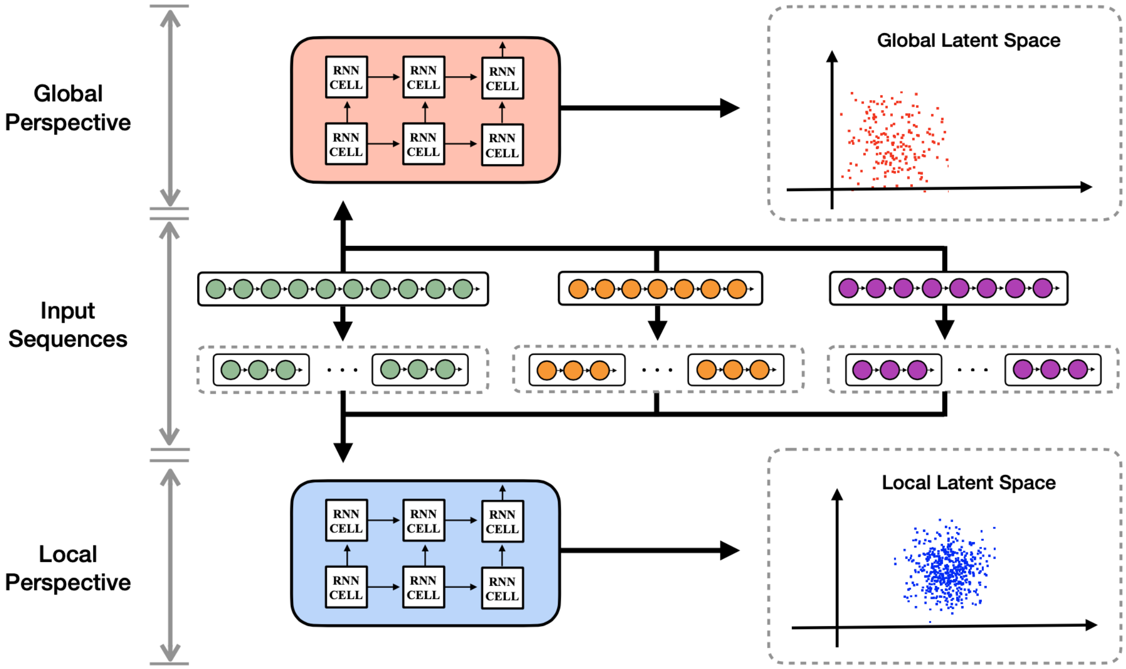


**Figure 2: The overview of the proposed framework OC4Seq. It consists of two major components that learn the sequential information from global and local perspectives, respectively.**

by other normal subsequences during the representation learning procedure. This could lead to a very high false negative rate.

## 4.3 Anomaly Detection from Local Perspective

In the previous subsection, we have introduced how to combine GRU and one-class classification objective to embed the normal sequences in a latent space where they lie within a small distance to a predefined center. However, local information that is vital for anomaly detection could be overwhelmed during this process. Thus, we design a subsequence learning component to detect the anomalies from the local perspective.

For a given event sequence, we construct subsequences of a fixed size $M$ with a sliding window. Therefore, each subsequence contains its unique local information, which plays an important role in determining whether the whole sequence is abnormal or not. To learn the representation of subsequence, we introduce the local GRU component that will model the sequential dependencies in every subsequence. Specifically, given a subsequence $\mathbf{x}^i_{t-M+1}, \mathbf{x}^i_{t-M+2}, \cdots, \mathbf{x}^i_t$ of length $M$, the local GRU processes the events sequentially and outputs $M$ hidden states, the last of which is used as the representation of the local subsequence:

$$\mathbf{h}^i_t = \mathrm{GRU}(\mathbf{x}^i_{t-M+1}, \mathbf{x}^i_{t-M+2}, \cdots, \mathbf{x}^i_t) \tag{10}$$

Thus, for all subsequences in a sequence, the GRU will obtain a sequence of hidden representations that encode the sequential dependencies in every local region as follows:

$$\mathbf{h}^i_1, \mathbf{h}^i_2, \cdots, \mathbf{h}^i_{N^i-M} = LocalGRU(\mathbf{x}^i_1, \mathbf{x}^i_2, \cdots, \mathbf{x}^i_{N^i}) \tag{11}$$

where LocalGRU is the name for the second GRU component that processes each subsequence.

For a normal event sequence, it is intuitive to assume that all of its subsequences are also normal. Thus, we further assume that all the local subsequences should be within a hypersphere in another latent space. To impose this assumption, we design the following objective function to guide the local sequence learning procedure:

$$\mathcal{L}_{local} = \min_{\Theta^L} \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N^i-M} \|\mathbf{h}_{N^i_j} - \mathbf{c}^L\|^2 + \lambda \|\Theta^L\|_F^2 \tag{12}$$

Here, $\mathbf{c}^L$ is a predefined center of another hypersphere in the latent space and $\mathbf{\Theta}^L$ contains all the trainable parameters of LocalGRU. Similarly, the first term penalizes the average distance between all normal subsequences to the center $\mathbf{c}^L$ and the second term is a regularizer.

## 4.4 The Objective Function and Optimization Procedure

In previous subsections, we have introduced the components of OC4Seq for detecting abnormal event sequences from both global and local perspectives, respectively. In this subsection, we design an objective function to combine them together. Specifically, given the global and local loss functions $\mathcal{L}_{global}$ and $\mathcal{L}_{local}$, the overall objective function of OC4Seq is defined as follows:

$$\min_{\Theta^L, \Theta} \mathcal{L} = \mathcal{L}_{global} + \alpha \mathcal{L}_{local} \tag{13}$$

where $\alpha$ is a hyper parameter that controls the contribution from local information in the sequence. This objective enables us to train the framework in an end-to-end manner. The specific optimization procedure is described next.

**Table 1: The key statistics of *RUBiS*, *HDFS*, and *BGL* datasets.**

| Dataset | # of normal | # of abnormal | # of log keys |
|---|---|---|---|
| RUBiS | 11,677 | 1,000 | 24 |
| HDFS | 558,221 | 16,838 | 28 |
| BGL | 9,543 | 985 | 1,540 |

**Optimization.** We use stochastic gradient descent (SGD) and its variants (*e.g.*, Adam) to optimize the objective function defined in Eq. (13). Following previous work [28], to accelerate the training process, the predefined centers $\mathbf{c}$ is computed as follows: given the untrained GRU, we firstly feed the sequences in the training set into it and obtain the sequence representation vectors. Then, we obtain an average vector by computing the mean value of all representation vectors and use it as $\mathbf{c}$. To obtain $\mathbf{c}^L$, a similar process is applied with untrained LocalGRU. Once $\mathbf{c}$ and $\mathbf{c}^L$ are obtained, their values will not be updated in the optimization process. The whole training process finishes when the objective value converges.
**Inference.** In the inference stage, for a given sequence, we calculate the loss defined in Eq. (13) as its anomaly score. The higher the value, the more likely the given sequence being an anomaly. We employ a simple threshold-moving algorithm [16] to determine the optimal threshold. Specifically, we first define a set of thresholds, and then utilize the validation set to evaluate our model's performance under each. The optimal threshold would be the one resulting in the best result based on the pre-defined evaluation metric. In this work, to locate the threshold with the optimal balance between precision and recall, we choose F1-score as the measurement, which is the harmonic mean of precision and recall, and is often considered as a more comprehensive evaluation metric for class imbalanced problem [22]. Note that depending on different application requirements, other metrics like Geometric Mean can also be used. When the anomaly score is higher than the chosen threshold, the sequence will be detected as an anomaly.

# 5 EXPERIMENT

In this section, we conduct extensive experiments to evaluate the proposed framework OC4Seq on three real-world system log datasets. We first describe the datasets and experimental settings. Then, we present the experimental results and observations. Finally, we conduct qualitative analysis to gain deep understandings of the proposed framework. To encourage reproducibility, we make our data and code publicly available[1].

## 5.1 Datasets

**RUBiS [2]:** This dataset is a weblog dataset and was generated by an auction site prototype modeled after eBay.com [2]. Specifically, each log message contains information related to a user web behavior including *user_id*, *date*, *request_info*, etc. Following previous work [37, 39], we first parse each log message into a structured representation, which consists of a log key and a variable among others. Next, the log keys of the same user are collected following the time order that forms an event sequence. Thus, each log key sequence represents a user behavior session on a web server. Each abnormal sequence corresponds to an attack case. In total, there are $11,677$ normal sequences and $1,000$ abnormal sequences.
**Hadoop Distributed File System (HDFS) [36]:** This dataset was generated by a Hadoop-based map-reduce cloud environment using benchmark workloads. It contains 11,175,629 log messages, of which 2.9% are labeled as anomalies by Hadoop experts. Each log message involves a block ID, a timestamp, and state information. To make the comparison fair, we used the publicly available dataset processed by [14]. As described in [14], the log messages are firstly parsed into structured text so that a log key is extracted from each log message. In addition, the log keys are sliced into sequences according to the associated block IDs. As a result, there are $558,221$ normal sequences and $16,838$ abnormal sequences.
**BlueGene/L (BGL) [27]:** This dataset contains $4,747,936$ log messages generated by a BlueGene/L supercomputer system with $131,072$ processors and $32,768$ GB memory at Lawrence Livermore National Labs. Each log message contains system information such as type, timestamp, nodes, content, etc. The log messages can be categorized into two types, *i.e.*, non-alert and alert. The non-alert messages are labeled as normal and alert messages are labeled as abnormal. The log messages are firstly parsed by Drain [17] Following previous work [26], the log keys are sliced using time sliding windows. A sequence is labeled as abnormal if it contains at least one abnormal message. After processing, there are $9,543$ normal sequences and $985$ abnormal sequences.

The statistics of the three datasets are summarized in Table 1. As stated in Section 2, this work focuses on the one-class/unsupervised setting, where the training dataset does not contain any abnormal sequence. Therefore, each dataset is split into training, validation, and test sets by the following process. Firstly, we randomly sample the training data from the normal sequence set. Then, we separately split the remaining normal sequences and all the abnormal sequences into validation and test sets with the ratio of 3/7 (validation/test). At last, we combine the two validation/test sets into one.

## 5.2 Baselines

We compare OC4Seq with the following five anomaly detection baselines:

**Principle Component Analysis (PCA)** [35]. PCA is a classic unsupervised method that has been extensively used for a variety of problems. More recently, it becomes a popular method for anomaly detection [36]. Specifically, it first constructs a count matrix $\mathbf{M}$, where each row represents a sequence, each column denotes a log key, and each entry $\mathbf{M}(i, j)$ indicates the count of $j^{th}$ log key in the $i^{th}$ sequence. Next, PCA learns a transformed coordinate system, where the projection lengths of normal sequences are small while the ones of abnormal sequences are large. Although it has been shown that PCA can be effective in detecting anomalies, especially in reducing false positives [14], it ignores the sequential information, which could play an important role in event sequence anomaly detection. We use the open-sourced implementation [18].

**Invariant Mining (IM)** [23]. IM is another popular unsupervised anomaly detection method. It is designed to automatically mine invariants in logs and assumes that the discovered invariants can capture the inherent linear characteristics of log flows.

**Table 2: The prediction performance comparison on *RUBiS*, *HDFS*, and *BGL* dataset.**

| Method | HDFS | | | RUBiS | | | BGL | | |
|---|---|---|---|---|---|---|---|---|---|
| | F1-score | Precision | Recall | F1-score | Precision | Recall | F1-score | Precision | Recall |
| OC-SVM | 0.509 | 0.622 | 0.431 | 0.351 | 0.220 | 0.869 | 0.336 | 0.215 | 0.764 |
| PCA | 0.634 | 0.968 | 0.471 | 0.784 | 0.862 | 0.718 | 0.423 | 0.269 | 0.993 |
| Invariant Mining | 0.943 | 0.893 | **1.000** | 0.912 | 0.841 | **0.996** | 0.428 | 0.273 | **1.000** |
| DeepLog | 0.941 | 0.952 | 0.930 | 0.935 | 0.885 | 0.992 | 0.326 | 0.196 | 0.980 |
| DeepSVDD | 0.872 | **0.964** | 0.796 | 0.970 | 0.974 | 0.966 | 0.410 | 0.298 | 0.659 |
| OC4Seq | **0.976** | 0.955 | 0.998 | **0.985** | **0.987** | 0.983 | **0.747** | **0.704** | 0.795 |

Similar to PCA, it firstly constructs a count matrix **M**. Next, IM learns sparse, integer-valued invariants with physical meanings from **M**. Finally, with the learned invariants, IM makes an invariant hypothesis. And the sequences that do not satisfy the hypothesis are detected as anomalies. As IM also relies on the $M$, it has similar drawbacks to PCA. The IM used in this work was implemented by [18].

**One-Class SVM (OC-SVM)** [29]. OC-SVM is a very effective one-class classifier that has been extensively used for anomaly detection [1, 21, 34]. Specifically, it learns a kernel that maps the normal data into a latent space, where all the normal sequence clusters in a small region. Thus, a sequence that does not belong to the cluster is regarded as abnormal. To apply OC-SVM, we first need to extract features from each sequence. In this work, we tried two models to extract features: sequence auto-encoder [10] and bag-of-words [38]. As we empirically found the latter often has better performance, we choose bag-of-words as the feature extractor. The OC-SVM used in this work was implemented with the scikit-learn package.

**DeepLog** [14]. DeepLog is a state-of-the-art log anomaly detection method. This method is based on an LSTM model, which tries to capture the sequential dependencies in sequences. Specifically, by training with normal sequences, it learns to predict the next token given the previously seen tokens in a sequence. During the test stage, for each time step in a sequence, DeepLog will output a probability distribution over all the log keys. If any of the actual tokens are not in the top $k$ candidates, it will regard the sequence as abnormal. Compared to other baselines, this method can utilize sequential information and has demonstrated state-of-the-art performance in previous work.

**DeepSVDD** [28]. DeepSVDD is a general one-class classifier that builds on deep neural networks. It simultaneously learns a representation vector for each data and optimizes the anomaly detection objective directly. To make the comparison fair, we use the same RNN models in OC4Seq as its representation learning component.

## 5.3 Experimental Settings

**Model Selection:** For all the methods with hyper-parameters, we use the validation set to select the best value and report the performance on the test set. For DeepLog, we follow the original paper's suggestion [14]. Specifically, both the $h$ and $g$ are selected from $\{8, 9, 10\}$, which denotes window size and candidate number, respectively. The number of layers is set to be 2 and the number of LSTM hidden units is 64. For OC4Seq, we use the same hyper-parameters as DeepLog and select $\alpha$ that controls the contribution of local subsequence from $\{0.01, 0.1, 1, 10\}$.

**Implementation Details:** We implemented OC4Seq with Pytorch 1.5. The model is trained using Adam [20] optimizer with the learning rate to be 0.01. The mini-batch size is chosen to be 64 and the model is trained for 100 epochs on a single NVIDIA GEFORCE RTX 2080 card.

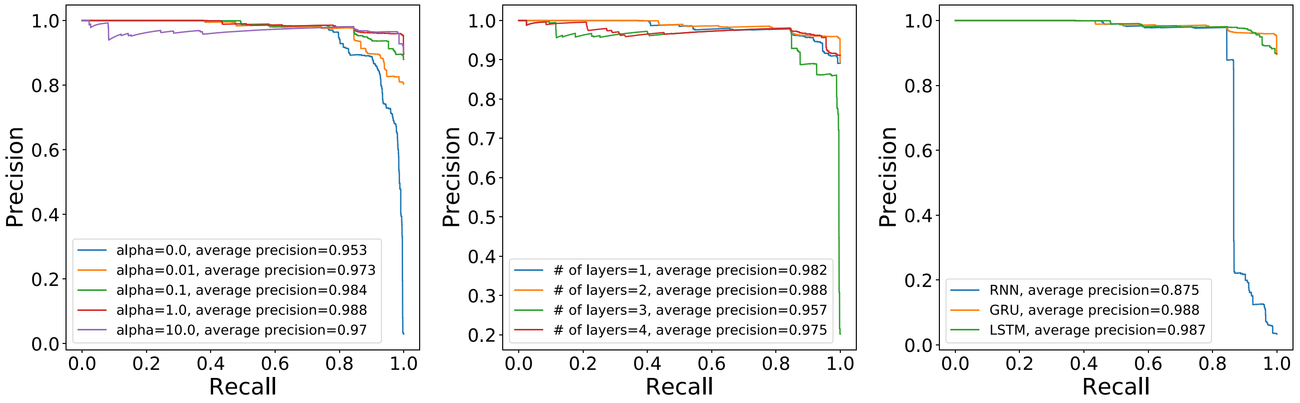
**Evaluation Metrics:** To measure the model performance on anomaly detection, we choose the widely-used Precision, Recall, and F1-score as the evaluation metrics. They are defined as follows:

$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN} \tag{14}$$
$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall}$$

where TP (True Positives) denotes the number of true abnormal sequences that are detected by the model, FP (False Positives) measures the number of normal sequences that are regarded as anomalies, and FN (False Negatives) denotes the number of abnormal sequences the model fails to report. Thus, based on these definitions, there is a well-known trade-off between precision and recall. On the other hand, the F1-score considers the balance of the two and is often considered as a more comprehensive evaluation metric.

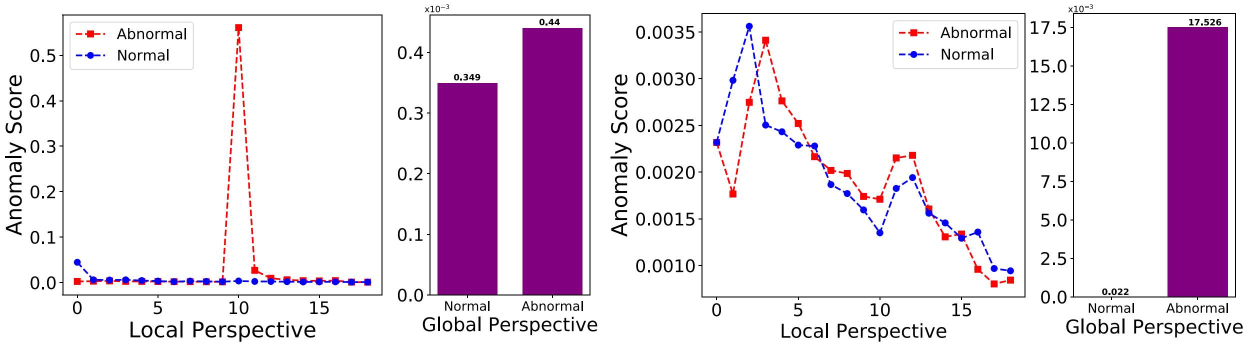
## 5.4 Performance Comparison

The results of all methods on three datasets are shown in Table 2. From the table, we can make the following observations: 1) On most datasets, OC-SVM has the worst performance. We argue that this is because OC-SVM is highly dependent on feature qualities. Unlike dense data where OC-SVM generally performs very well, it is very hard to extract meaningful features from discrete event sequences. 2) IM and DeepLog outperform PCA significantly in most of the evaluation metrics. This is expected as both IM and DeepLog are designed specifically for anomaly detection for log data. 3) IM and DeepLog generally have comparable results in terms of F-1 score. It is interesting to observe that IM always achieves a very impressive Recall while DeepLog is better in Precision. We argue that this difference could be caused by the fact that DeepLog focuses much on the local subsequence information while IM always concentrates on global sequence information. 4) The performance of DeepSVDD varies significantly in different datasets. This is because it only focuses on global information in the sequences. Thus, it can do very well with the datasets where local information is not important and becomes much worse otherwise. 5) On all the



**(a) Precision-Recall curves with different alpha.** **(b) Precision-Recall curves with # of layers.** **(c) Precision-Recall curves with different RNN cell types.**

**Figure 3: Validation Precision-Recall curves on HDFS dataset.**



**(a) Showcase of the importance of local information.** **(b) Showcase of the importance of global information.**

**Figure 4: The local and global anomaly scores of HDFS log key sequences.**

datasets, the proposed framework OC4Seq has achieved the best F-1 scores. In addition, when comparing to the second-best method, the performance gain brought by OC4Seq is significant. In terms of Precision, OC4Seq still achieved the highest value in most cases. For Recall, OC4Seq was only slightly outperformed by IM which has much lower precision scores. 6) All of the methods performed much worse on BGL datasets than the other two datasets. This is because BGL involves much more log keys that can make the task extremely difficult. Moreover, it is interesting to note that the DeepLog method becomes especially ineffective as it heavily relies on the next key prediction which is very difficult when log keys space becomes large. In this challenging case, the improvement from OC4seq over other baselines becomes even more remarkable.

As a summary, from the experimental results on three datasets, our proposed framework OC4Seq demonstrates its superior performance over the representative baselines. We argue this is because OC4Seq can capture sequential information from both local subsequence and whole sequence and it directly optimizes the anomaly detection objective. Next, we design further experiments to gain deeper understandings of OC4Seq.

## 5.5 Parameter Analysis

In this subsection, we analyze key hyper-parameters and components of OC4Seq. We only report the performance on the HDFS validation dataset as we have similar observations on the others. Moreover, the performance is evaluated by the Precision-Recall curve as it eliminates the need to choose a specific anomaly threshold and very suitable for datasets with imbalanced label distribution. We also report the area under the Precision-Recall curve (average precision) where the higher the value, the better the performance.

We firstly vary the value of $\alpha$ from $\{0, 0.01, 0.1, 1, 10\}$, which controls the contribution from local subsequence information. The results are shown in Figure 3a. From the figure, we can see that with the increase of $\alpha$, the performance firstly increases and then decreases. The initial increase demonstrates the importance to incorporate local subsequence information while the latter decrease suggests that the global sequential information is also very essential and should not be overwhelmed by local information.

Next, we vary the number of RNN layers from $\{1, 2, 3, 4\}$, and the results are shown in Figure 3b. These results suggest that more layers do not necessarily lead to better performance as it may cause other issues such as overfitting. Thus, it is important to select a proper value through the validation process.

Finally, to investigate how different types of RNN cells affect anomaly detection performance, we experienced on popular cells, *i.e.*, RNN (Vanilla), GRU, and LSTM. The results are shown in Figure 3c. From the figure, it is easily seen that the LSTM and GRU cells have very similar performance while vanilla RNN achieves much worse results. These results are consistent with previous work [9]. Due to its simpler structure, we choose GRU in OC4Seq.

## 5.6 Case Study

In this subsection, to further understand how the local and global information contribute to anomaly detection, we conduct case studies involving two pairs of representative log key sequences from the HDFS dataset. Specifically, for a sequence, we use the trained model to calculate the anomaly scores of each subsequence and the whole sequence. The higher the anomaly score is, the more likely the sequence is abnormal. The results are shown in Figure 4.

In Figure 4a, we show the first pair of normal and abnormal sequences. The left panel uses dot lines to demonstrate the anomalous scores for local subsequences. Each dot denotes one anomalous score (y-axis) of $x^{th}$ subsequence (x-axis). The right panel uses the bar to show the anomalous score for the whole sequence (global information). From the figure, we observe that the two sequences have comparable anomalous scores for the whole sequence. Thus, it is very difficult to detect the abnormal one purely from the global perspective. However, from the local perspective, we can see that the $10^{th}$ subsequence of the abnormal sequence has a very high anomalous score while the anomalous scores of the subsequence of the normal sequence are all very low. Therefore, in this case, the local information plays a very important role in detecting anomalies.

In Figure 4b, the anomalous scores of the second pair of sequences are shown. Unlike the previous case, the second pair of sequences has very similar anomalous scores for local subsequences. This makes it hard to detect anomalies from the local perspective. However, the abnormal sequence has a significantly higher anomalous score from the global perspective than the normal one. Therefore, the global information contributes a lot to detect the anomaly in this case. From the two cases, we further illustrate the importance of combining both local and global information in a sequence for anomaly detection.

## 5.7 Visualization of Normal and Abnormal Sequences

To gain insights into the one-class classifier objective function, we project the global representations of both normal and abnormal sequences in the HDFS validation set to a two-dimensional space by Local Linear Embedding techniques [13]. The visualization of the two-dimensional space is shown in Figure 5. We observe that the normal sequences generally cluster together and lie in a very small region. On the other hand, the abnormal sequences spread all over the place. Therefore, the visualization clearly shows that by directly minimizing the anomalous score, which measures the distance between a normal data point and a center point, the one-class classifier objective function is very effective to guide the model to separate the abnormal and normal sequences in the latent space.

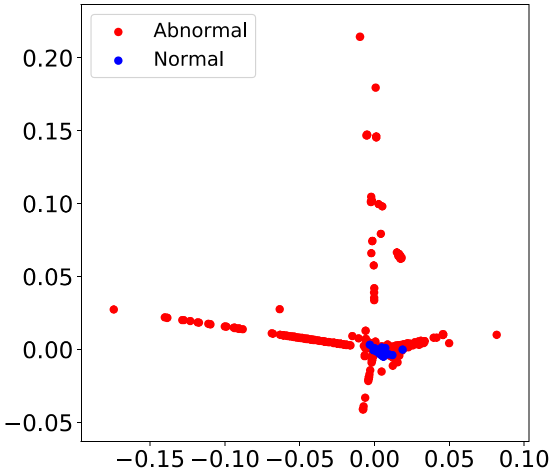


**Figure 5: The visualization of the HDFS dataset. Abnormal and normal data are denoted by red and blue dots, respectively.**

## 6 RELATED WORK

In this section, we briefly review the related work on discrete event sequence anomaly detection and one-class classifier.

**Anomaly Detection for Discrete Event Sequence:** One group of traditional methods for event sequence anomaly detection is based on similarity measurement [3, 4], where the similarity between a test sequence and training sequences is calculated and the anomalies are detected based on similarity scores. One major issue with similarity based methods is that there is a lack of intrinsic measurement of similarity between discrete event sequences and the effectiveness of the model can be largely affected by the choice of the similarity measurement. Other popular traditional methods include Markovian techniques and Hidden Markov Model based ones [8, 30]. However, the capacities of these models are too small to capture the complex long-term dependencies in the sequences. More recently, neural network based methods [14, 32] have been proposed for anomaly detection and achieved great success due to their strong representation learning ability and large modeling capacities.

**One-Class Classifier for Anomaly Detection:** One-class classifiers focus on learning the properties of objects from a single class and are trained to detect objects that are from any other classes. Therefore, they are very suitable for anomaly detection tasks and have gained great attention [29, 31, 34]. The two most successful one-class classifiers are OC-SVM and SVDD [29, 31] that are based on traditional kernel tricks. Only recently, the deep neural network based one-class classifier is proposed by Ruff *et al.* [28]. Inspired by SVDD [31], the authors designed a novel one-class classification objective that has very nice theoretical properties and can effectively train CNNs for image anomaly detection tasks. Considering the success of one-class classifiers, we also build our framework upon similar objective functions. However, unlike previous works that mainly focus on dense data such as images, we conduct pioneering research of building one-class recurrent neural networks for discrete event sequences, which have very unique challenges comparing to the dense data.

## 7 CONCLUSION

In this paper, we propose OC4Seq, an end-to-end one-class recurrent neural network for discrete event sequence anomaly detection. OC4Seq can deal with multi-scale sequential dependencies and detect anomalies from both local and global perspectives. Specifically, OC4Seq incorporates an effective anomaly detection objective that can guide the learning process of sequence models. The trained multi-scale sequence model in OC4Seq explicitly maps the local subsequence and whole sequence into different latent spaces, where the normal data points are enclosed by hyperspheres with minimum volume. The proposed OC4Seq has consistently shown superior performance than representative baselines in extensive experiments on three real-world datasets. In addition, through parameter analysis and case studies, the importance of capturing multi-scale sequential dependencies for discrete event sequence anomaly detection has been well demonstrated. Moreover, the visualization of the sequence representations qualitatively suggests the effectiveness of anomaly detection objectives.

## 8 ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable suggestions and comments. We thank Jiaping Gui for the inspiring discussion and dataset. Zhiwei Wang and Jiliang Tang are supported by the National Science Foundation (NSF) under grant numbers CNS1815636, IIS1928278, IIS1714741, IIS1845081, IIS1907704, and IIS1955285.

## REFERENCES

[1] Mennatallah Amer, Markus Goldstein, and Slim Abdennadher. 2013. Enhancing one-class support vector machines for unsupervised anomaly detection. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description.*

[2] Cristiana Amza, Emmanuel Cecchet, Anupam Chanda, Alan L Cox, Sameh Elnikety, Romer Gil, Julie Marguerite, Karthick Rajamani, and Willy Zwaenepoel. 2002. Specification and implementation of dynamic web site benchmarks. In *5th Workshop on Workload Characterization.*

[3] Shyam Boriah, Varun Chandola, and Vipin Kumar. 2008. Similarity measures for categorical data: A comparative evaluation. In *Proceedings of the 2008 SIAM International Conference on Data Mining.* 243–254.

[4] Suratna Budalakoti, Ashok N Srivastava, and Matthew Eric Otey. 2008. Anomaly detection and diagnosis algorithms for discrete symbol sequences with applications to airline safety. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 39, 1 (2008), 101–113.

[5] Cheng Cao, Zhengzhang Chen, James Caverlee, Lu-An Tang, Chen Luo, and Zhichun Li. 2018. Behavior-Based Community Detection: Application to Host Assessment In Enterprise Information Networks. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management.* 1977–1985.

[6] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2010. Anomaly detection for discrete sequences: A survey. *TKDE* 24, 5 (2010), 823–839.

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[8] Sung-Bae Cho and Hyuk-Jang Park. 2003. Efficient anomaly detection by modeling privilege flows using hidden Markov model. *Computers & Security* 22, 1 (2003), 45–55.

[9] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[10] Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems.* 3079–3087.

[11] Boxiang Dong, Zhengzhang Chen, Hui Wang, Lu-An Tang, Kai Zhang, Ying Lin, Zhichun Li, and Haifeng Chen. 2020. Anomalous Event Sequence Detection. *IEEE Intelligent Systems* (2020).

[12] Boxiang Dong, Zhengzhang Chen, Hui (Wendy) Wang, Lu-An Tang, Kai Zhang, Ying Lin, Zhichun Li, and Haifeng Chen. 2017. Efficient Discovery of Abnormal Event Sequences in Enterprise Security Systems. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management.* 707–715.

[13] David L Donoho and Carrie Grimes. 2003. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences* 100, 10 (2003), 5591–5596.

[14] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 CCS.* 1285–1298.

[15] Manish Gupta, Jing Gao, Charu C Aggarwal, and Jiawei Han. 2013. Outlier detection for temporal data: A survey. *TKDE* 26, 9 (2013), 2250–2267.

[16] Haibo He and Yunqian Ma. 2013. *Imbalanced learning: foundations, algorithms, and applications.* John Wiley & Sons.

[17] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R Lyu. 2017. Drain: An online log parsing approach with fixed depth tree. In *2017 IEEE International Conference on Web Services (ICWS).* IEEE, 33–40.

[18] Shilin He, Jieming Zhu, Pinjia He, and Michael R Lyu. 2016. Experience report: System log analysis for anomaly detection. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE).* IEEE, 207–218.

[19] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[20] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[21] Kun-Lun Li, Hou-Kuan Huang, Sheng-Feng Tian, and Wei Xu. 2003. Improving one-class SVM for anomaly detection. In *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics*, Vol. 5. IEEE, 3077–3081.

[22] Rushi Longadge and Snehalata Dongre. 2013. Class imbalance problem in data mining review. *arXiv preprint arXiv:1305.1707* (2013).

[23] Jian-Guang Lou, Qiang Fu, Shengqi Yang, Ye Xu, and Jiang Li. 2010. Mining Invariants from Console Logs for System Problem Detection.. In *USENIX Annual Technical Conference.* 1–14.

[24] Chen Luo, Zhengzhang Chen, Lu-An Tang, Anshumali Shrivastava, Zhichun Li, Haifeng Chen, and Jieping Ye. 2018. TINET: learning invariant networks via knowledge transfer. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 1890–1899.

[25] Mirco Marchetti and Dario Stabili. 2017. Anomaly detection of CAN bus messages through analysis of ID sequences. In *2017 IEEE Intelligent Vehicles Symposium (IV).* IEEE, 1577–1583.

[26] Weibin Meng and *et al.* 2019. Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *IJCAI-19*, Vol. 7. 4739–4745.

[27] Adam Oliner and Jon Stearley. 2007. What supercomputers say: A study of five system logs. In *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07).* IEEE, 575–584.

[28] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. 2018. Deep one-class classification. In *International Conference on Machine Learning.* 4393–4402.

[29] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural Computation* 13, 7 (2001), 1443–1471.

[30] Xiaobin Tan and Hongsheng Xi. 2008. Hidden semi-Markov model for anomaly detection. *Appl. Math. Comput.* 205, 2 (2008), 562–567.

[31] David MJ Tax and Robert PW Duin. 2004. Support vector data description. *Machine Learning* 54, 1 (2004), 45–66.

[32] Aaron Randall Tuor, Ryan Baerwolf, Nicolas Knowles, Brian Hutchinson, Nicole Nichols, and Robert Jasper. 2018. Recurrent neural network language models for open vocabulary event-level cyber anomaly detection. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence.*

[33] Shen Wang, Zhengzhang Chen, Xiao Yu, Ding Li, Jingchao Ni, Lu-An Tang, Jiaping Gui, Zhichun Li, Haifeng Chen, and Philip S. Yu. 2019. Heterogeneous Graph Matching Networks for Unknown Malware Detection. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence.* 3762–3770.

[34] Yanxin Wang, Johnny Wong, and Andrew Miner. 2004. Anomaly intrusion detection using one class SVM. In *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.* IEEE, 358–364.

[35] Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems* 2, 1-3 (1987), 37–52.

[36] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I Jordan. 2009. Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles.* 117–132.

[37] Xiao Yu, Pallavi Joshi, Jianwu Xu, Guoliang Jin, Hui Zhang, and Guofei Jiang. 2016. Cloudseer: Workflow monitoring of cloud infrastructures via interleaved logs. *ACM SIGARCH Computer Architecture News* 44, 2 (2016), 489–502.

[38] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. 2010. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics* 1, 1-4 (2010), 43–52.

[39] Xu Zhao, Kirk Rodrigues, Yu Luo, Ding Yuan, and Michael Stumm. 2016. Non-intrusive performance profiling for entire software stacks based on the flow reconstruction principle. In *12th {USENIX}-({OSDI}*.