IEICE TRANSACTIONS

on Information and Systems

VOL. E102-D NO. 9
SEPTEMBER 2019

A PUBLICATION OF THE INFORMATION AND SYSTEMS SOCIETY
The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

IEICE TRANS. INF. & SYST., VOL.E102–D, NO.9 SEPTEMBER 2019

PAPER Special Section on Log Data Usage Technology and Office Information Systems

Latent Variable Based Anomaly Detection in Network System Logs*

Kazuki OTOMO[†a)], **Satoru KOBAYASHI**[††b)], ***Nonmembers***, **Kensuke FUKUDA**[††,†††c)], ***and*** **Hiroshi ESAKI**[†d)], ***Members***

SUMMARY System logs are useful to understand the status of and detect faults in large scale networks. However, due to their diversity and volume of these logs, log analysis requires much time and effort. In this paper, we propose a log event anomaly detection method for large-scale networks without pre-processing and feature extraction. The key idea is to embed a large amount of diverse data into hidden states by using latent variables. We evaluate our method with 12 months of system logs obtained from a nation-wide academic network in Japan. Through comparisons with Kleinberg's univariate burst detection and a traditional multivariate analysis (i.e., PCA), we demonstrate that our proposed method achieves 14.5% higher recall and 3% higher precision than PCA. A case study shows detected anomalies are effective information for troubleshooting of network system faults.

key words: *network operation, system logs, syslog, anomaly detection, latent variable analysis, variational autoencoder*

1. Introduction

System logs are one of the most useful sources to understand the state of a network. In an operational network, syslog is widely used for collecting network logs and allows one to gather logs from all devices at one server. In network operation, appearing unusual logs is a signal for strange or unexpected behavior. Thus, we consider such appearance of the unexpected logs as an anomaly. For example, if one finds a large number of BGP status change logs for a particular AS in a day, compared to those in the past days, one interprets this as instability of the route to that AS.

However, it is not easy for network operators to investigate the details of network problems with the logs because of their diverse and massive nature. Many log analysis methods for finding anomalies and their root causes have been proposed to overcome this problem [2]–[7]. In many cases, one first classifies the logs by their message type (i.e., *log template*) then treats them as statistical time series to be later processed through statistical analysis. System logs have an intrinsic nature that makes automatic analysis very difficult; their time series show a variety of characteristics such as periodicity, burstiness, randomness, and sparseness, compared with other time series data. Most studies have been devoted to developing effective pre-processing methods (e.g., filtering, smoothing, denoising) and appropriate discriminative log time series features for analyzing their own data. Applying statistical algorithms with these features, they detect anomalies (e.g., burstiness, change points) in log time series. However, these existing approaches have two limitations. First, we require enough domain knowledge about the underlying networks for optimizing the analysis methods. Second, we need to rely on predefined thresholds for using predefined features even though behavior of log time series is various and different for each dataset.

To detect anomalies (i.e., unexpected behavior in log time series), we propose a robust statistical method that handles complicated system log time series, without any specific (case-by-case) preprocessing (e.g., filtering, smoothing, auto-regression) for log anomaly detection. The key idea of our approach is to embed a large amount of diverse data into hidden states by using latent variables and detect anomalies in latent space (as shown in Fig. 1). A latent variable analysis is also known as topic modeling or Latent Dirichlet Allocation (LDA) often used in natural language processing. A topic model discovers topics as latent variables from a collection of documents. We borrow the idea of this approach: We first translate raw log messages into log

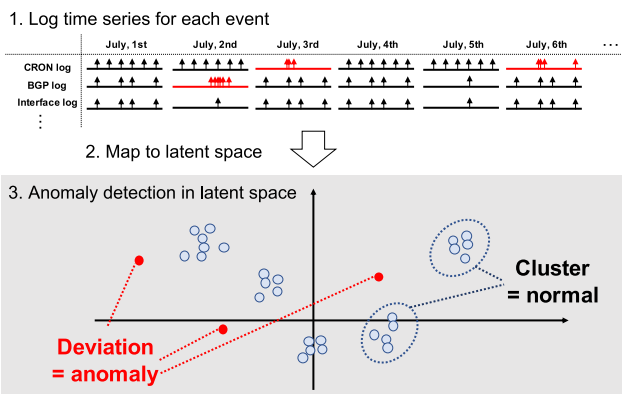

Fig. 1 Key idea of our proposed framework

Manuscript received November 12, 2018.
Manuscript revised April 2, 2019.
Manuscript publicized June 7, 2019.

[†]The authors are with Graduate School of Information Science and Technology, the University of Tokyo, Tokyo, 113–8654 Japan.
[††]The authors are with the National Institute of Informatics, Tokyo, 101–8430 Japan.
[†††]The author is with Department of Informatics, Sokendai, Tokyo, 101–8430 Japan.

*This paper is an extended version of work published in Ref. [1]

a) E-mail: otomo@hongo.wide.ad.jp
b) E-mail: sat@nii.ac.jp
c) E-mail: kensuke@nii.ac.jp
d) E-mail: hiroshi@wide.ad.jp

DOI: 10.1587/transinf.2018OFP0007

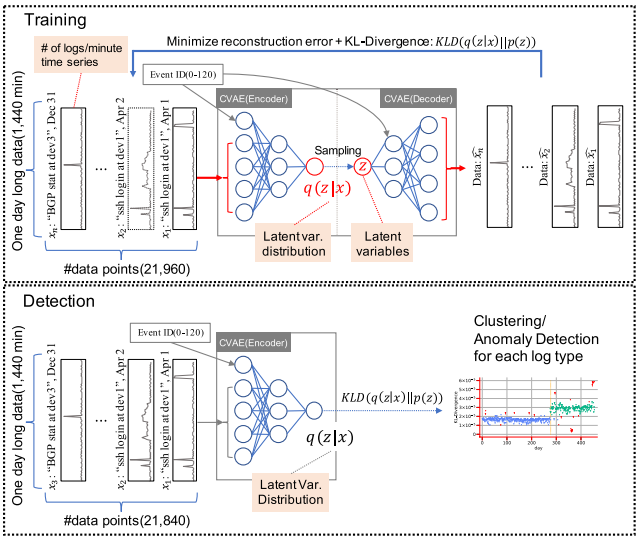

Fig. 2 Methodology overview

time series for each type of logs (step (1) in Fig. 1). Next, log time series are mapped into latent variables per day per type of logs (step (2) in Fig. 1). In the latent space, values of latent variables represent the trend of the corresponding time series. After that, we apply a clustering method to latent variables and detect deviations from detected clusters in the latent space (step (3) in Fig. 1). We consider clusters in the latent space are groups of normal trend time series in the real log time series. On the other hand, deviations from clusters in the latent space are interpreted as anomalies because the deviation in the latent space means deviated time series trend in the real log time series. Thus, with latent variables, time series can be represented by their trend without any domain knowledge or preprocessings to real log time series.

Our proposed method consists of two phases (see also Fig. 2): (1) The training phase embeds log time series characteristics into latent variables by using Conditional Variational Autoencoder (CVAE). To mitigate the difficulty in handling various time series from many devices together, we provide CVAE with type of log messages as a conditional label. (2) The detection phase highlights anomalies deviating from the distribution of the latent variables with clustering algorithms.

Using syslog data collected in a nation-wide academic network in Japan (SINET4), we confirm through evaluation that CVAE has higher discriminative power for analyzing complex time series than Kleinberg's univariate burst detection [8] and a traditional multivariate analysis (Principal Component Analysis; PCA). With a case study, we demonstrate that our method is useful in troubleshooting network system faults.

The contribution of this paper is twofold. We first propose our latent variable-based method for highly diverse log time series data without any data-specific preprocessing. Next, we discuss the effectiveness of the method when we applied it to real network syslog data.

2. Related Work

Many studies have been conducted for finding anomalies and their root causes [2], [4], [5] in log data. Zhong et al. [3] proposed an anomaly detection method for both device and network errors with fine log time series feature creation. Kimura et al. [9] introduced an online failure prediction method based on log time series features. Lu et al. [6] focused on the task duration time and proposed root cause analysis methods with distributed computing system logs. As these methods are based on data specific feature creation, they perform well in each considered environment. However, to apply these methods to other network systems, we have to re-define features to optimize these methods. This requires deep domain knowledge of the underlying network systems to make efficient use of these methods. In addition, these methods miss unknown or new anomalies, which are not captured by the pre-defined features. Instead, we focus on an approach that does not require pre-definition of anomalies but learns the normal state of the system from log data.

There are methods of giving new insights for operators with knowledge mining from log data [10], [11]. Kobayashi et al. [7] proposed a time series causal inference method in network logs. Hacker et al. [12] introduced a log classification method based on the severity of network operation. These methods do not require predefined features because the features characterizing the data can be obtained through learning. However, in contrast with our method, they do not aim at anomaly detection but knowledge mining.

Recently, some studies have applied deep learning techniques to anomaly detection in logs [13]–[16]. Du et al. [17] proposed a log anomaly detection method by using Long-Short Term Memory (LSTM). They built LSTM models for each type of logs and whole log time series in application level logs (OpenStack and HDFS logs). However, we have to handle network logs which are larger amount and more diverse than application level logs. Therefore, we circumvent anomaly detection directly from dataset but try to reduce the dimension of input dataset by latent variable analysis.

We focus on anomaly detection without specific feature creation for more general analysis of logs. PCA is a standard algorithm of learning or obtaining time series features from data. Lakhina et al. [18] proposed a general traffic analysis method based on feature learning with PCA. They successfully detected traffic anomalies without specific definitions of those anomalies. However, as log data are highly sparse and discrete, which differ with traffic data, it is not enough to learn features with such a naive approach (i.e., PCA). Thus, we use CVAE, which is based on a higher assumption that observed data are subject to latent variables. We choose a PCA based method as a baseline algorithm and aim to outperform it with our proposed method.

3. Methodology

3.1 Overview

The key idea of our proposed method is to model log time series characteristics based on the latent variables. Latent variable analysis is conducted to attempts to explain observable data by unobservable (i.e., latent) variables. In our context, as the observable data are log time series, we intend to represent the characteristics of observable log time series, such as periodicity, frequency, burstiness, by latent variables. By applying a clustering algorithm or anomaly detection method to the latent variables, we can distinguish anomalous behavior of log time series from the normal behavior. Formally, as it is difficult to compute latent variables directly, we have to estimate them by assuming certain statistical models. In this study, we rely on Conditional Variational Autoencoder (CVAE) for estimating latent variables.

An overview of the proposed method is illustrated in Fig. 2. First, we construct the time series of the number of log appearances per device per template for one day. We call it a *data point*. Next, with CVAE, we reduce the dimensions of the original vectors by using CVAE. We call such a reduced vector a latent variable. We train CVAE model so that the latent variable effectively represents the potential behavior of log time series. Then, each data point is embedded into a latent variable. After that, we apply a clustering algorithm to the latent variables and finally detect outliers in log time series as anomalies.

3.2 Training Phase

We briefly explain CVAE, a variation of the Variational Autoencoder (VAE). The VAE [19] is a stochastic variational inference method based on the variational Bayesian approach. Since this is an unsupervised method, annotations for logs are not necessary.

Let us consider one-day long log time series $\mathbf{x}$ generated by a random process based on invisible continuous random variable $\mathbf{z}$. This $\mathbf{z}$ is also subject to a prior distribution $p_\theta(\mathbf{z})$. Thus, $\mathbf{x}$ is subject to a conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$. The goal is to obtain latent variables $\mathbf{z}$ from input $\mathbf{x}$. In order to get the latent variables, we estimate the distribution $p_\theta(\mathbf{z}|\mathbf{x})$ with Bayes' theorem and approximate distribution $q_\phi(\mathbf{z}|\mathbf{x})$. Now, the objective is to maximize the marginal likelihood (MLH) $\log p_\theta(\mathbf{x}) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) + \mathcal{L}(\theta, \phi, \mathbf{x})$. This equation is rearranged as follows (details are given in [19], [20]).

$$\log p_\theta(\mathbf{x}) \geq \mathcal{L}(\theta, \phi, \mathbf{x}), \tag{1}$$

$$\mathcal{L}(\theta, \phi, \mathbf{x}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]. \tag{2}$$

We train the model that generates $q_\phi(\mathbf{z}|\mathbf{x})$ from input $\mathbf{x}$ by optimizing the lower bound $\mathcal{L}(\theta, \phi, \mathbf{x})$. The training process is as follows. First, we consider the prior distribution as a Gaussian $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$ and let $q_\phi(\mathbf{z}|\mathbf{x})$ be a multivariate Gaussian. Next, we estimate the parameters (ϕ) of the posterior distribution ($q_\phi(\mathbf{z}|\mathbf{x})$) with a fully connected neural network (encoder). Then, we acquire the distribution of the latent variable $q_\phi(\mathbf{z}|\mathbf{x})$, and through the sampling process from it, we obtain the latent variable $\mathbf{z}$. Finally the decoder neural network reconstructs $\hat{\mathbf{x}}$ from the latent variable $\mathbf{z}$, and we feedback the loss values in Eq. (2). Reviewing Eq. (2), we can consider the first term (Kullback-Leibler divergence, KLD) as the distance between estimated distributions $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{z})$, and the second term (Marginal Likelihood, MLH) as the reconstruction error between $\mathbf{x}$ and $\hat{\mathbf{x}}$. Through these processes, we can compute Eq. (2) and proceed with the training of the encoder and decoder neural networks.

The CVAE [20] is a variation of the VAE. CVAE uses label information when generating the latent variable $\mathbf{z}$ and reconstructing $\hat{\mathbf{x}}$. We define the conditional label as the event that is an index of a unique combination of devices and log templates.

3.3 Detection Phase

After applying CVAE to log time series data, we obtain the latent variables for each data. Now, as the latent variables well-describe time series trends, the goal of this phase is to build upon the time series description by latent variables to find the deviation from "normal" states.

Some latent analysis methods define the anomaly level based on reconstruction errors [18], [21]. In our case, we can use MLH for reconstruction errors. However, there are two problems with using MLH. (1) The reconstruction process is stochastic and MLH values follow a Gaussian distribution. (2) MLH is biased by the intensity of the original data.

To solve these problems, we rely on the distance between latent variable distribution ($q_\phi(\mathbf{z}|\mathbf{x})$) and the assumed prior distribution ($p_\theta(\mathbf{z})$) by computing $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))$ in Eq. (2). After training, the learned model generates the distribution of the latent variable ($q_\phi(\mathbf{z}|\mathbf{x})$) from input data $\mathbf{x}$ as the output in a hidden layer (shown at the bottom of Fig. 2). We use this distribution $q_\theta(\mathbf{z}|\mathbf{x})$ for detection, not the sampled value $\mathbf{z}$. Then, we deterministically compute the KLD ($D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))$). The KLD represents the trend in the input time series $\mathbf{x}$. Thus, input data with similar KLD value belong to the same trend. Therefore, we can obtain reference behavior by clustering KLD values. Note that the KLD value is not an anomaly level but a state of input time series. Thus, we consider the data whose KLD value deviates from others as an anomaly regardless of the magnitude of numeric value of the KLD.

Figure 3 shows an example of the KLD and MLH values of the dataset. The horizontal axis shows data points in descending order of the number of log appearances, and the vertical axis shows the KLD (top) and MLH (bottom) values. In particular, the gray marks indicate days without any log appearances (i.e., 1,440-long zero vector), and the blue ones show more than equal to one log appearances on that

day. Even if we input no log appearances data (i.e., gray marks in the Figure) to CVAE, KLD values are different for each event thanks to the weight of the conditional label (i.e., event ID). It enables such data points to belong to appropriate clusters depending on the behavior of the event. As shown in the figure, the MLH values are more spread and biased by the number of log appearances. In addition, they increase along with the number of log appearances. On the other hand, the KLD values are not biased by the number of log appearances and are more stable than MLH values. Therefore, we use the KLD as an indicator of time series trend.

Figure 4 is a cumulative distribution of normalized KLD values (blue curve) and normalized MLH values (orange curve). Red lines show 99% points in KLD and MLH. As shown in this figure, KLD requires the range from 10^{-7} to 10^{-2} to represent 99% of data points. However, MLH needs 10 times wider range than KLD (from 10^{-7} to 10^{-1}). Thus, KLD is more suitable for the indicator of time series trend because dense data is easier to handle than sparse data.

Once latent variable distribution is obtained and the KLD is computed one can choose any clustering method for anomaly detection. In this study, we used a well-known density-based clustering algorithm, Density-Based Algorithm for Discovering Clusters (DBSCAN) [22]. By applying DBSCAN to the KLD time series for each event, we detect clusters of KLD (i.e., normal state of that type of log) and anomalies that do not belong to any clusters.

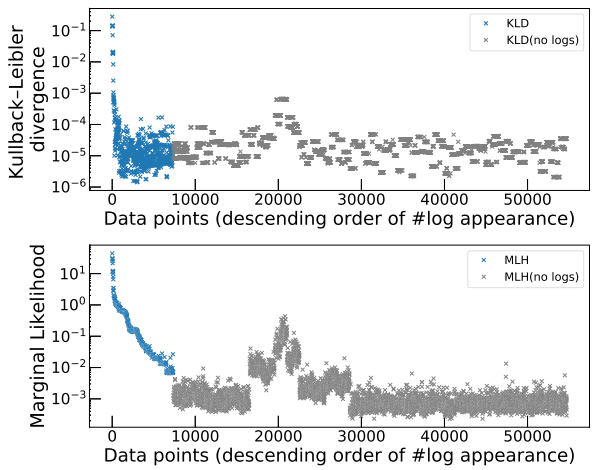

Fig. 3 Kullback Leibler divergence and Marginal Likelihood

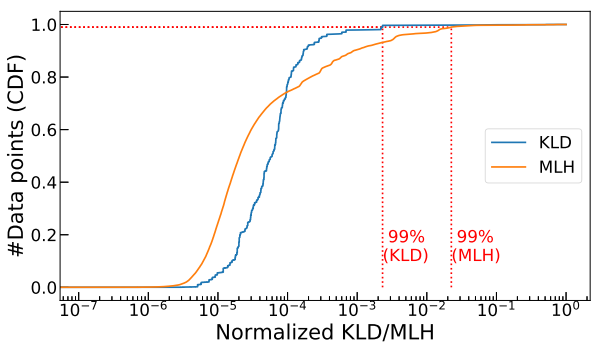

Fig. 4 Kullback Leibler divergence and Marginal Likelihood (CDF)

4. Dataset and Comparison Algorithms

4.1 Dataset

We use a set of network logs collected at SINET [23], a Japanese research and education network. This network connects over 800 academic organizations in Japan and consists of eight core routers, 50 edge routers, and 100 layer-2 switches. We selected a part of 365 days data from 2012/4/1 to 2013/3/31 gathered from commodity L2 switches and L3 routers. Table 2 lists well-known services working in this network.

As log messages are string data, statistical techniques cannot be applied directly. Thus, we generate log templates from raw log messages with the supervised learning approach proposed by Kobayashi et al. [7]. Log templates are log messages without variables (e.g., IP address), as shown in Fig. 5. We then classify logs into log templates and extract time series data from their time stamps for each template per device. We thus generate 1,789 unique log templates from the whole dataset. We define an *event* as 365 of 1-day log time series, which have the same log template in one device. For our analysis, we manually selected 120 events and applied CVAE to them (shown in Table 1) to make sure that the dataset has several anomalies and the diversity of the time series. We confirm that these selected events include a large variety of time series trends such as periodicity, burstiness, and sparseness as well as diverse categories of log templates shown in Fig. 6.

We construct log time series of the number of log

Table 1 SINET4 Dataset

	Training	Testing
logs	174,623	284,317
devices	24	24
templates	33	33
events	120	120
data points	21,960	21,840
start	'12/04/01	'12/10/01
end	'12/09/30	'13/03/31

Table 2 Examples of network services in SINET4

Routing	BGP, OSPF
Management	ssh
Monitor	SNMP, ARP
Network	LACP, MTU, TCP
Service	NTP
VPN	L2VC, MPLS

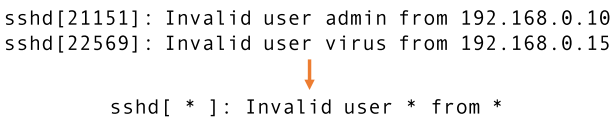

Fig. 5 An example of log templates

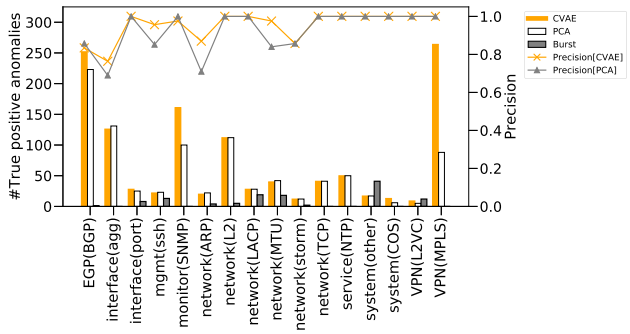

Fig. 6 Number of Detected Anomalies and Their Precision Per Log Category

appearances for each minute per event per day (i.e., data point). The processing flow is as follows: After we classified raw log data into the type of logs per device, one event (for example, "*ssh login at device 1*") has 365 days long log entries. We first count the number of log appearances for each minute. Then, we split them for each days. We now have 365 data points of event "*ssh login at device 1*". In other words, each data point consists of one time series; It has 1,440 minute-length as X-axis and the number of log appearances as Y-axis (shown as x_1 and x_2 time series graphs in Fig. 2). We also use conditional labels built on event labels concatenating log time series. To input event labels into CVAE, we first assign unique IDs to all the events and then encode them to one-hot vectors. If the "*ssh login at device 1*" event has ID i, the conditional label is a 120 long vector and values are zero except index i. The value of index i is one. Then, the conditional label has 120 dimensions. After repeating the same process to all the 120 events, we finally obtain $120 \times 365 = 43{,}800$ data points and conditional labels.

In the end, we explain training and testing data for our analysis. The first three months of the data include a lot of unusual behavior because this period was the migration and beginning of network operation in SINET4. Thus, we exclude the data in the first three months (10,920 data points in total) and use the rest of 43,800 data points. Then, we split this dataset into training dataset and testing dataset (see also Table 1). The training dataset is the first half of dataset (21,960 data points, that is from 2012/4/1 to 2012/9/30) and the testing dataset is the second one (21,840 data points, that is from 2012/10/1 to 2013/03/31).

4.2 Baseline Algorithms

To understand the effectiveness of CVAE for log anomaly detection, we compare it to two traditional baseline algorithms: Kleinberg's burst detection [8] and PCA. The three algorithms are summarized in Table 3.

4.2.1 Burst Detection

Otomo et al. [24] conducted log analysis focusing on the burstiness and causality of log time series. They first removed trivial logs (e.g., periodic and very frequent logs) to prevent detecting trivial bursts then detected log bursts with Kleinberg's burst detection [8]. Applying the same algorithm to our dataset, we confirm 188 log bursts out of all 21,840 data points. In the following comparison, we use these bursts as a part of reference of log anomalies.

Table 3 Comparison of methods

name	multivariate	determin.	linear trans.
burst detection	no	yes	-
PCA	yes	yes	yes
CVAE	yes	no	no

4.2.2 PCA

To detect traffic anomalies, Principal Component Analysis (PCA) is a well-studied algorithm [18], [21]. It translates a set of input traffic time series data into main and residual subspaces with a linear transformation. This algorithm is similar to CVAE mapping raw data to the latent space, though CVAE is a non-linear transformation. We implement a PCA-based anomaly detector. Applying PCA to a set of log time series, we obtain principal components of the input dataset and separate them into main and residual principal components with a threshold based on their variance coverage. We then reconstruct each time series with the main components and compute the reconstruction errors with the original data. Finally, we apply a clustering method (DBSCAN) to these reconstruction errors and obtain anomalies.

5. Evaluation

In this section, we discuss the evaluation of our method. We use 365 day-long system log data obtained from SINET4 and detect anomalies per day. First, we compare the performance of our method with PCA and burst detection technique using 6 month-long training dataset and 6 month-long testing dataset. Next, we evaluate dependency of training data set size on the accuracy of anomaly detection. Finally, we present a case study that demonstrates the effectiveness of our method for network troubleshooting.

5.1 Parameter Tuning

We first briefly describe the parameter tunings of CVAE. Encoder and decoder networks in CVAE each have four hidden layers. The input layer has 1,560 dimensions (1,440 dimension of time series and 120 dimension of label data). The encoding neural network has four layers, and each layer has 512, 256, 128 and 64 units in order from the input layer. We set the latent variables dimensions to 10. The decoding neural network has the opposite structure of the encoder, but the output size is 1,440 so that the output is a reconstructed time series. To avoid over fitting, we dropout 30% of units for each layer during training. During training, we empirically

set the number of epochs to 50 based on test trial results. We confirm that loss values converge after training. Next, we compute KLD for each time series using the learned encoder network and apply DBSCAN to KLD for each event. We tune DBSCAN parameters for each event so that a size of cluster is longer than a week. As the training process is stochastic, the results are not the same even if the loss values are converged. To mitigate this uncertainty, we train the model ten times and adopt anomalies detected in majority votes.

We use a commodity computer (Xeon CPU and GTX-1080 GPU) for processing. One training takes 489.5s on average of 10 trials. The detection phase requires 45.1s on average to calculate KLD and conduct DBSCAN.

5.2 Result Overview

For better understanding detected anomalies in network management, we manually inspected all the test data points and annotated them as anomaly or normal based on the SINET4 trouble ticket data and our domain knowledge for network management; For example, "*ssh login*" logs have periodic appearance in our data because they are generated by a CRON process including hourly remote ssh access. In this case, we consider periodic appearance as "normal" and lack of periodicity as "anomaly". We use these annotations as ground truth.

We first show that CVAE outperforms PCA (Table 4). Note that CVAE's results are averages of 6 trials because our proposed method is stochastic. CVAE detects more number of anomalies with higher accuracy than PCA; 1,215 anomalies with CVAE and 1,044 with PCA. CVAE's recall is about 14.5% higher than PCA's recall and CVAE's precision is about 3% higher than PCA's one. The number of overlap anomaly was 898.

We also confirm that CVAE and PCA detected ≈ 90% of the bursts detected by Kleinberg's burst detection algorithm as shown in the last row in Table 4. By inspecting missing bursts, we find two parameter tuning issues: (1) When similar burst patterns last for a few days, DBSCAN extracts them as a cluster, not outliers. The algorithm thus does not detect these bursts as anomalies. (2) Intensive bursts cause CVAE and PCA to detect a relatively small burst as a normal.

Figure 6 is a histogram of detected anomalies per log category. Orange bars indicate true positive anomalies detected with CVAE, white ones with PCA, and gray ones with burst detection. Orange line shows CVAE precision and white one shows PCA precision. In most cases, we find that CVAE correctly detects anomalies such as burstiness and lack of periodicity. Focusing on the case in which PCA detects more true positive anomalies than CVAE (e.g., interface(agg)), we find CVAE has a higher precision than PCA thanks to the CVAE's robustness against noise (see § 5.4 (2)). On the other hand, when CVAE detects more true positive anomalies than PCA (e.g., VPN(MPLS), monitor(SNMP), EGP(BGP)), we find that they still keep high precision thanks to its robustness against outliers (see § 5.4 (3)). Therefore, our proposed method is better balanced between the number of detected anomalies and precision than the traditional PCA-based method. Note that a detected anomaly means time series behavior deviating from the normal state as defined in §1.

Table 4 Summary of results

	PCA	CVAE
Data	21,840	21,840
Anomalies	1,044	1,215
Precision	88.4%	91.5%
Recall	59.4%	74.0%
#Bursts	169 (89.9%)	170 (90.0%)

5.3 Dependency of Training Data Size

We also evaluate the performance of our proposed method using different training data size: 1 - 6 month-long. As shown in Table 1, we have 6 month-long training dataset and 6 month-long testing dataset, which are continuous in time series. We use different size of training dataset in the order of newest date: 1 month-long dataset is from '12/9/1 to '12/9/30, 2 month-long dataset is from '12/8/1 to '12/9/30, and so forth. Figure 7 shows detailed results for each size of training data. Blue lines are CVAE's results and orange ones are PCA's results. CVAE's results are averages of 3 trials in this figure. As shown in Fig. 7, CVAE's precision and recall outperform PCA's for each training data size. Focusing on the number of false positives (shown on the right in the figure), CVAE's one increases in less than 3 month-long training data. In the worst case, the number of false positives is larger than PCA's one in 1 month-long training data. Thus, we conclude that CVAE requires 3 month or more training data for obtaining appropriate results.

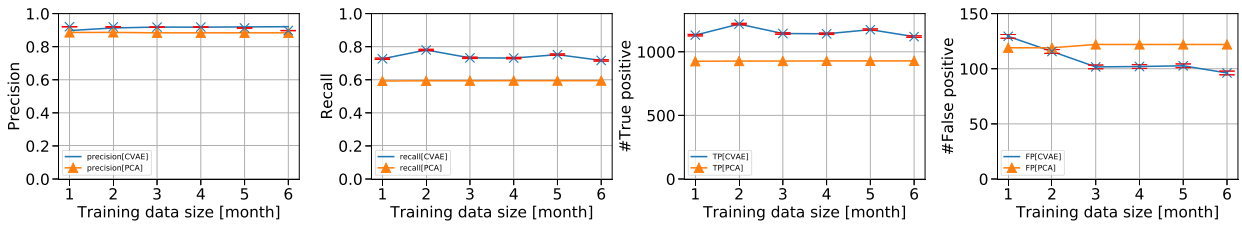

Fig. 7 Performance dependency on different training data size: Precision, Recall, #True positive and #False positive, from left to right, respectively.

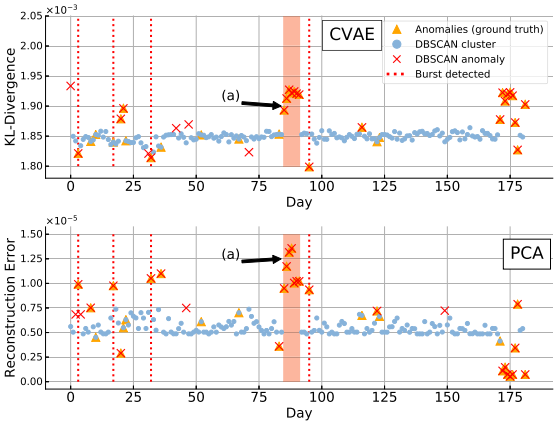

Fig. 8 Comparison between KLD (CVAE), reconstruction errors (PCA) and burst detection results

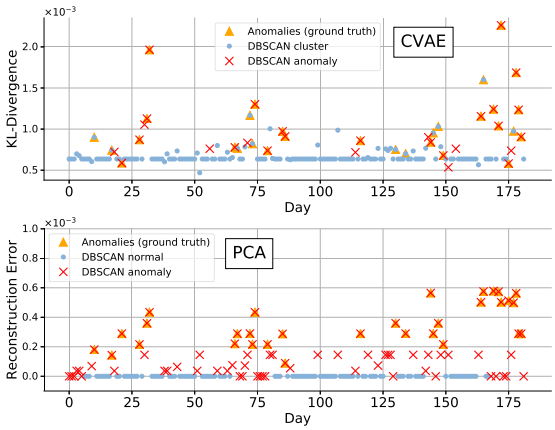

Fig. 9 Robustness of CVAE against noisy logs

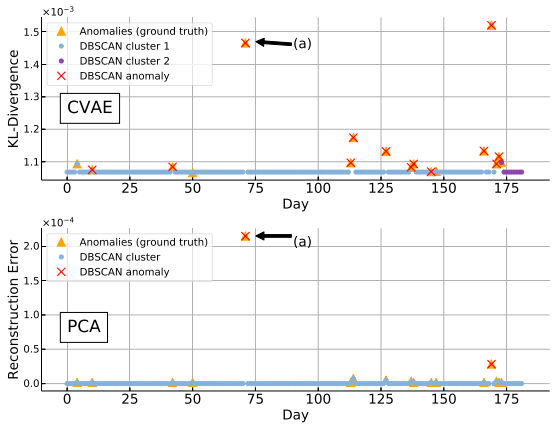

Fig. 10 Robustness of CVAE against large outliers

5.4 Detailed Comparison

(1) Anomalies hidden in periodicity:

Figure 8 shows the detailed results of periodic remote access logs for 182 days. The top graph shows KLD computed with CVAE and the bottom graph illustrates reconstruction errors obtained with PCA. Red dotted lines indicate anomalies detected by burst detection. These logs appear once per hour due to an automated remote monitoring script. We confirm significant outliers in the figure due to missing periodic logs (label (a)). As non-periodic anomalies represent failures of automatic remote access login, we intend to detect these non-periodic anomalies in addition to burst anomalies. As expected, the burst detection finds only burst anomalies. As shown in this figure, CVAE and PCA correctly detect these two type of anomalous data from others. Thus, CVAE and PCA correctly learn event-wise features (in this case, periodic appearance) thanks to their discriminative power.

(2) Robustness against noisy logs:

If there are noisy logs without outliers, PCA yields many false anomalies due to its noise-sensitivity. However, as CVAE is stochastic and merges the multiple results, we can mitigate the effect of noises. Figure 9 (interface aggregate warning event) shows an example of this case. The top graph shows KLD in CVAE, and the bottom graph shows the reconstruction errors in PCA. The solid blue circles are normal points and the crosses are anomalies based on DBSCAN clustering. The orange triangles are true anomalies. This event has discreteness in the number of log appearance because its logs always appear with multiple lines (i.e., #network interfaces in its device). As shown in the figure, PCA is affected by this trend and also shows discreteness in reconstruction error. This trend makes anomaly detection or clustering difficult because errors spread with constant interval. With CVAE, however, the KLD values are relatively spread compared with reconstruction errors, and DBSCAN detects other anomalies.

(3) Robustness against outliers:

KLD is robust against the intensity (i.e., number of log appearances) of the original data, as shown in Fig. 3. On the other hand, we confirm that PCA's reconstruction errors (not shown in the figure) are also biased to the intensity, similar to the case with MLH.

Figure 10 (MPLS path down event) shows an example of CVAE's sensitivity to a small number of log appearances. The top graph shows KLD with CVAE, and the bottom graph shows the reconstruction errors with PCA. The solid blue circles are normal points and the crosses are anomalies based on DBSCAN clustering. Note that anomalies in the top graph are clustering results after merging ten trials, but the KLD values are a result of one trial. These data have one large outlier (around day 71; label (a)) which has a larger number of log appearances than other days. As shown in the figure, PCA is affected by this outlier, and many anomalous points are not detected. With CVAE, however, the KLD values are relatively spread compared with reconstruction errors, and DBSCAN detects other anomalies. However, we also confirm that less than 1% of the data have high KLD values ($> 10^{-4}$), as shown in Fig. 3. It is still possible that the clustering process may miss small KLD value changes

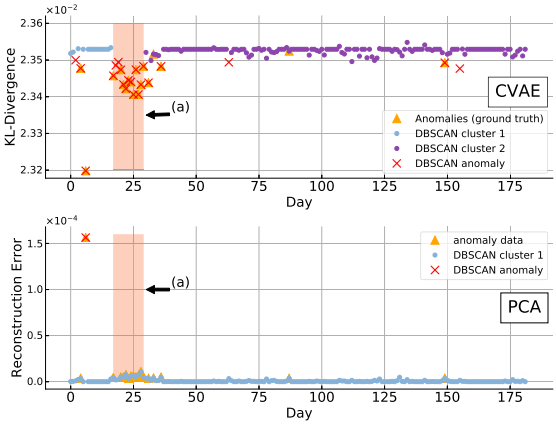

Fig. 11 Case study: BGP neighbor state change logs

due to these high values. We will work to improve this issue by tuning the clustering process as future work.

Therefore, CVAE exhibits better robustness to outliers and noises than PCA.

5.5 Case Study: BGP State Change

We now discuss anomaly examples detected from our proposed method in Fig. 11. In this figure, each mark shows the KLD of the BGP state change logs from a router per day. Almost all days have a baseline ($\approx 2.35 \times 10^{-2}$). Checking the raw logs around the 25th day (a red area labeled (a) in the figure), we find that a connection with one particular AS suddenly became unstable. These failures had been also reported in a trouble ticket. Thus, they are useful information for network operators or troubleshooting. As shown in the bottom figure, PCA fails to detect most of these anomalies due to a larger outlier. CVAE, on the other hand, successfully discovered them.

6. Conclusion

We proposed a log analysis method based on the latent variable model to detect network system anomalies. We used Conditional Variational Autoencoder (CVAE) and applied a clustering algorithm to log time series with KL-divergence of the latent variable distribution. We confirmed that this method works well for highly sparse and diverse time series through comparison with Kleinberg's burst detection and a traditional PCA-based method. Our method achieved 14.5% higher recall and 3% higher precision than a PCA-based method. We also evaluated performance dependency of different training data size and concluded 3 month or larger size of training data is sufficient for our proposed method. In addition, our case study showed that some detected anomalies are helpful in finding network troubles. Towards the deployment of our method to real network management, we will work on addressing how to update the trained model with upcoming syslog data.

Acknowledgements

The authors would like to thank the SINET operation team for providing the syslog and trouble tickets data. The authors would also like to thank Johan Mazel and anonymous reviewers for their comments on this paper. This work is supported by JSPS KAKENHI Grant Number JP19K20262, and the MIC/SCOPE #191603009.

References

[1] K. Otomo, S. Kobayashi, K. Fukuda, and H. Esaki, "Finding anomalies in network system logs with latent variables," Proc. ACM SIGCOMM BIGDAMA'18, pp.8–14, 2018.
[2] E. Baseman, S. Blanchard, and E. Zongzelimyuntedu, "Relational Synthesis of Text and Numeric Data for Anomaly Detection on Computing System Logs," Proc. IEEE ICMLA'16, pp.2–5, 2016.
[3] J. Zhong, W. Guo, and Z. Wang, "Study on network failure prediction based on alarm logs," Proc. ICBDSC'16, pp.23–29, 2016.
[4] M. Moh, S. Pininti, S. Doddapaneni, and T.-S. Moh, "Detecting Web Attacks Using Multi-stage Log Analysis," Proc. IEEE IACC'16, pp.733–738, 2016.
[5] M. Shatnawi and M. Hefeeda, "Real-time failure prediction in online services," Proc. IEEE INFOCOM'15, pp.1391–1399, 2015.
[6] S. Lu, B. Rao, X. Wei, B. Tak, L. Wang, and L. Wang, "Log-based Abnormal Task Detection and Root Cause Analysis for Spark," Proc. IEEE ICWS'17, pp.389–396, 2017.
[7] S. Kobayashi, K. Otomo, K. Fukuda, and H. Esaki, "Mining causality of network events in log data," IEEE TNSM, vol.15, no.1, pp.53–67, 2018.
[8] J. Kleinberg, "Bursty and Hierarchical Structure in Streams," Proc. ACM KDD'02, pp.91–101, 2002.
[9] T. Kimura, A. Watanabe, T. Toyono, and K. Ishibashi, "Proactive failure detection learning generation patterns of large-scale network logs," Proc. CNSM'15, pp.8–14, 2015.
[10] Z. Zheng, Z. Lan, B.H. Park, and A. Geist, "System log pre-processing to improve failure prediction," Proc. IEEE DSN'09, pp.572–577, 2009.
[11] T. Kimura, K. Ishibashi, T. Mori, H. Sawada, T. Toyono, K. Nishimatsu, A. Watanabe, A. Shimoda, and K. Shiomoto, "Spatio-temporal factorization of log data for understanding network events," Proc. IEEE INFOCOM'14, pp.610–618, 2014.
[12] T. Hacker, R. Pais, and C. Rong, "A markov random field based approach for analyzing supercomputer system logs," IEEE Trans. Cloud Comput., pp.1–1, 2017.
[13] R. Vinayakumar, K.P. Soman, and P. Poornachandran, "Long short-term memory based operation log anomaly detection," Proc. ACM ICACCI'17, pp.236–242, Sept. 2017.
[14] M. Nadeem, V. Nigam, D. Anagnostopoulos, and P. Carretas, "Automating network error detection using long-short term memory networks," arXiv, 2018.
[15] W. Meng, Y. Liu, S. Zhang, D. Pei, H. Dong, L. Song, and X. Luo, "Device-agnostic log anomaly classification with partial labels," arXiv, 2018.
[16] T. Tan, S. Gao, W. Yang, Y. Song, and C. Lin, "Two new term weighting methods for router syslogs anomaly detection," Proc. IEEE HPCC'16, pp.1454–1460, Dec. 2016.
[17] D. Min, L. Feifei, Z. Guineng, and S. Vivek, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," Proc. ACM CCS'17, pp.1285–1298, 2017.
[18] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," Proc. ACM SIGCOMM'04, vol.34, no.4, p.219, 2004.
[19] D.P. Kingma and M. Welling, "Auto-Encoding Variational Bayes,"

arXiv, no.Ml, pp.1–14, 2013.
[20] D.P. Kingma, D.J. Rezende, S. Mohamed, and M. Welling, "Semi-Supervised Learning with Deep Generative Models," arXiv, pp.1–9, 2014.
[21] W. Xu, L. Huang, A. Fox, D. Patterson, M.I. Jordan, L. Huang, A. Fox, D. Patterson, and M.I. Jordan, "Detecting Large-Scale System Problems by Mining Console Logs," Proc. ACM SOSP'09, pp.117–131, 2009.
[22] M. Ester, H.P. Kriegel, J. Sander, and X. Xu, "A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," Proc. ACM KDD'96, pp.226–231, 1996.
[23] S. Urushidani, M. Aoki, K. Fukuda, S. Abe, M. Nakamura, M. Koibuchi, Y. Ji, and S. Yamada, "Highly available network design and resource management of SINET4," Telecommunication Systems, vol.56, no.1, pp.33–47, 2014.
[24] K. Otomo, S. Kobayashi, K. Fukuda, and H. Esaki, "An Analysis of Burstiness and Causality of System Logs," Proc. AINTEC'17, pp.16–23, 2017.

Kazuki Otomo is a Master's student at the Graduate school of Information Science and Technology, the University of Tokyo. His research interest includes knowledge extraction in network time series.

Satoru Kobayashi received the Ph.D degree in information science and technology from the University of Tokyo, Tokyo Japan, in 2018. He is a project postdoc researcher with the National Institute of Informatics (NII). His research interests are network management and data mining.

Kensuke Fukuda received the Ph.D. degree in computer science from Keio University, Kanagawa Japan, in 1999. He is an Associate Professor with the National Institute of Informatics (NII) and the Graduate University for Advanced Studies (SOKENDAI). His research interests span Internet traffic analysis, anomaly detection, modeling networks and QoS over the Internet.

Hiroshi Esaki received the Ph.D. degree in electronic engineering from the University of Tokyo, Tokyo, Japan, in 1998. He is a Professor at the Graduate school of Information Science and Technology, the University of Tokyo. Currently, he is Vice president of JPNIC, the Director of the WIDE Project, and the Board of Trustees of the Internet Society.