The Chinese University of Hong Kong
Department of Computer Science and Engineering

Term Paper (2020-21, 2nd term)

Cover Sheet

Title: CMAnomaly: Collaborative Machine-based Multivariate KPIs Anomaly Detection

Name: Jinyang Liu Student I.D.: 1155154504

Contact Tel. No.: (+86) 13609752831 Email A/C: jyliu@cse.cuhk.edu.hk

Supervisor(s): Prof. Michael R. Lyu

Committee Chairman (1st Internal Examiner): Prof. Ho-fung Leung

2nd Internal Examiner: Prof. Hong Xu

Mode of Study: ☐ Part-time ☑ Full-time

Term of study: Term 2, 2020-21

Thesis Proposal: ☐ Yes ☑ No

(If you are going to have candidacy exam, please click "yes")

CMAnomaly: Collaborative Machine-based Multivariate KPIs Anomaly Detection

LIU, Jinyang

Supervised by

Prof. Michael R. Lyu

Computer Science and Engineering
The Chinese University of Hong Kong
April 22, 2021

Abstract

As modern software systems continue to grow in terms of complexity and volume, anomaly detection on multivariate Key Performance Indicators (KPIs), which profile systems' health status, becomes more and more critical and challenging. In particular, the dependency between KPIs and their historical patterns plays a critical role in pursuing prompt and accurate anomaly detection. Existing approaches fall short of industrial needs for being unable to capture such information explicitly and efficiently. To fill this significant gap, in this paper, we propose CMAnomaly, an anomaly detection framework on multivariate KPIs based on collaborative machine. Remarkably, the proposed collaborative machine is a mechanism to capture the pairwise KPI interactions along with feature and temporal dimensions with linear time complexity. Cost-effective models can then be employed to leverage both the dependency between KPIs and their historical patterns for anomaly detection. The proposed framework is extensively evaluated with both public data and industrial data collected from a large-scale online service system of Huawei Cloud. The experimental results demonstrate that compared with state-of-the-art baseline models, CMAnomaly achieves an average F1 score of 0.9494, outperforming baselines by 4.45%, and runs $10\times \sim 20\times$ faster. Furthermore, our framework has been successfully deployed in industrial practice.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As modern software systems, e.g., online service systems, have grown to an unprecedented scale, failures become inevitable, leading to significant revenue loss and user dissatisfaction [4, 5, 10, 11, 28]. To understand the health status of a system, Key Performance Indicators (KPIs) such as network traffic, server response delay, and CPU usage rate should be closely monitored. Anomaly detection over the KPIs is an important means to ensure the reliability and availability of the system, which aims to discover unexpected events or rare items in data. Many efforts [2, 8, 19, 25] have been devoted to univariate KPI anomaly detection, which deals with only a single type of KPI. However, more often than not, different types of KPIs collectively can indicate the occurrence of anomalies more precisely. Different system components (e.g., microservices, servers) are tightly coupled, and failures tend to trigger anomalies in multiple KPIs. For example, a problematic load balance server is often accompanied by a burst on both round-trip delay and in-bound traffic rate, which will further increase CPU utilization. However, a sudden rise in CPU utilization alone could be caused by regular service upgrades, which should not be regarded as an anomaly. Compared to univariate KPI, anomaly detection on multivariate KPIs is more challenging. We have

identified three main reasons:

Demanding industrial requirements. In modern software systems, minutes of downtime could cause an expensive drain on company revenue. Therefore, suspicious anomalies should be quickly identified. Moreover, as systems continuously undergo feature upgrades and system renewal, the patterns of multivariate KPIs may shift accordingly. To accommodate such ever-changing pattern shifts, the anomaly detection model must support fast model retraining.

Large-volume and noisy data. Although terabytes and even petabytes of KPI data are being generated everyday, most of them do not contain much valuable information. For example, a significant portion of KPIs only records plain system runtime behaviors. Moreover, a low signal-to-noise ratio is also a critical issue. Thus, multivariate KPI anomaly detection with the presence of noise is challenging.

Sparsity of KPIs' dependency. In industrial systems, hundreds or even thousands of KPIs are being monitored. The dependencies among KPIs are very sparse, i.e., most KPIs are not or weakly dependent on other KPIs. Therefore, how to automatically learn the dependencies among different KPIs is critical towards efficient multivariate KPI anomaly detection.

In the literature, many studies have shifted to anomaly detection on multivariate KPIs, which mainly resorts to different neural network models. For example, OmniAnomaly [23] proposes to learn the normal patterns of multivariate time series by modeling data distribution through stochastic latent variables. Anomalies are then determined by reconstruction probabilities. Similarly, Malhotra et al. [17] used an LSTM-based (long short-term memory) encoder-decoder network to learn time series's normal patterns and Zhang et al. [27] used an attention-based convolutional LSTM network. Although tremendous progress has been made, we still observe two major limitations

of existing approaches: 1) the interactions among KPIs are not explicitly modeled, and 2) the efficiency falls behind industrial needs. Specifically, previous approaches [17, 18, 23] detect anomalies on multivariate KPIs mainly by stacking different types of KPIs into a feature matrix and feeding it to sophisticated neural network models. In this manner, each KPI corresponds to one feature, and KPIs' interactions are implicitly learned at the cost of complicated model design, resulting in inevitable false alarms. More recent studies tackle this problem by constructing an $m \times m$ KPI inner-product matrix [27] or a complete graph [29] for m different KPIs to capture the pairwise KPI interaction, both of which yield an $\mathcal{O}(m^2)$ computation complexity. Similarly, heavy models such as graph neural networks [21] are employed to learn KPIs' dependencies by feeding the matrix or training on the graph. Different from them, we argue that by properly modeling the KPIs interactions, cost-effective neural network models can be leveraged for anomaly detection.

In this paper, to overcome the aforementioned limitations, we propose CMAnomaly, an efficient unsupervised model for anomaly detection over multivariate KPIs. Specifically, we propose a collaborative machine to conduct cross-feature and cross-time KPI interactions by taking the inner product of pairwise KPIs and pairwise timestamps. Moreover, to capture the existing sparse dependency, we assign a weight to each KPI (or timestamp), and the importance of a pair of dependencies is measured by the product of corresponding weights. However, such a weighted interactions mechanism still costs quadratic computation as previous approaches. To alleviate this problem, inspired by factorization machines [20], we reformulate the proposed collaborative machine and reduce the computation from quadratic time complexity to linear complexity, which ensure the efficiency of CMAnomaly. After that, to consider both the feature and temporal factors simultaneously, the results of interactions from both aspects are concatenated and fed to a cost-

effective multi-layer perceptron (MLP). Particularly, the model is trained by learning from the historical normal patterns of KPIs and predicting their future states. Anomalies are then identified based on prediction errors, which has been proven to be effective in many related studies [12, 29].

To sum up, in this paper, we make the following major contributions:

- We propose CMAnomaly, an efficient unsupervised model for anomaly detection over multivariate KPIs. CMAnomaly explicitly models the interactions of KPIs along both temporal and feature dimensions. More importantly, we reduce the computation of our model from quadratic complexity to linear complexity. In doing this, anomaly detection can be performed efficiently to process a large volume of data in the real world.

- We conduct extensive experiments with both public and industrial data collected from Huawei Cloud. The experimental results demonstrate that CMAnomaly achieves 0.9494 F1 score with 4.45% improvement over the state-of-the-art approaches. Moreover, CMAnomaly can run $10\times \sim 20\times$ faster than the baseline models.

- We have successfully incorporated CMAnomaly into the troubleshooting system of the company Huawei Cloud. Feedback from on-site engineers confirms its practical benefits conveyed to various online services and products. We also share the success story to benefit the community.

The remainder of this paper is organized as follows. Chapter 2 introduces the background and problem statement of this paper. Chapter 3 describes the proposed methodology. Chapter 4 shows the experiments and the results. Chapter 5 discusses the related work. Chapter 6 presents our success story and lessons learned from practice. Finally, Chapter 7 concludes this work.

Chapter 2

Background

2.1 Dependency of Multivariate KPIs

In recent years, the complexity and scale of modern software systems are growing at a rapid speed. Various types of KPIs that profile an entity's health status need to be closely monitored [23]. Particularly, an entity can be a logical one like a service or a physical one like a computing server. They provide system operators and engineers with the most prompt and direct way to understand the system. As different system components are tightly-coupled and work collaboratively, there exists some dependency among KPIs, i.e., Multivariate KPIs. Consequently, when failures are encountered, the dependent KPIs are likely to exhibit abnormal behaviors simultaneously. Such dependencies reflect some intrinsic properties of the system and thus provide more reliable evidence for multivariate KPI anomaly detection.

A real-world example is provided in Figure 2.1, which is from a public dataset released by [23]. The four KPIs in the figure constitute a typical set of dependent KPIs. Particularly, *CPU LOAD* and *ETH INFLOW* are strongly correlated as their curves exhibit a highly similar trend. Without considering such dependency, we will miss a complete picture of the system's health

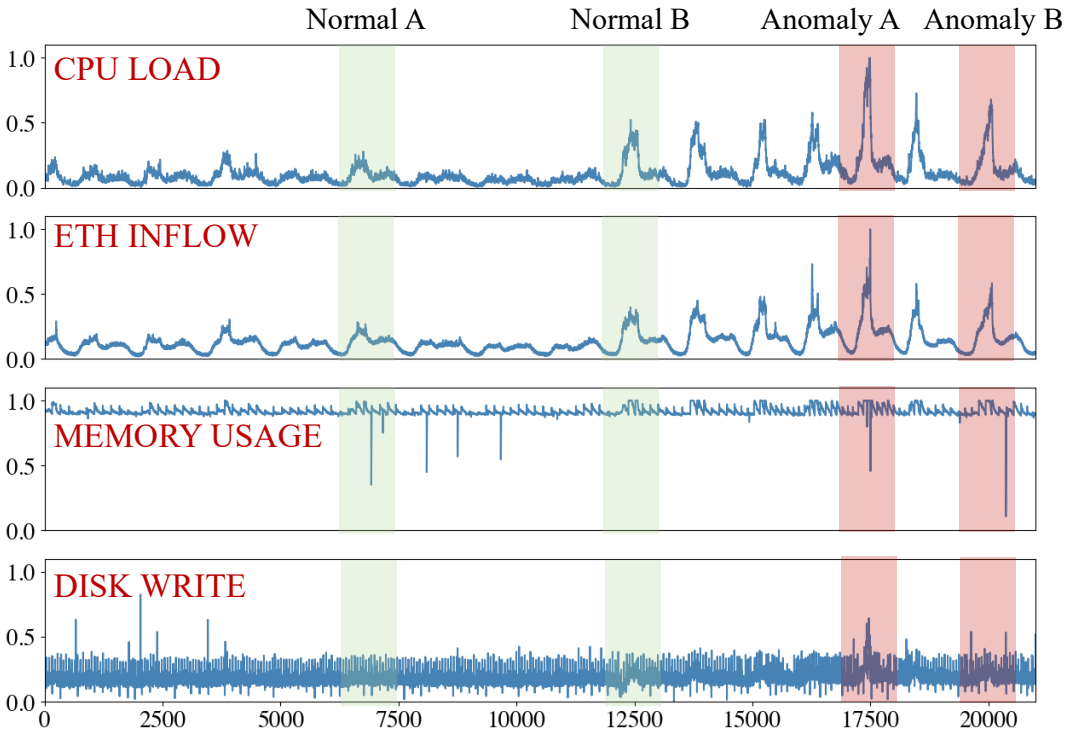

Figure 2.1: Multivariate KPIs Snippet from Server Machine Dataset

status. For example, in the segment marked as *Normal A*, we can see a clear spike in *MEMORY USAGE*, which would be flagged as an anomaly without a glance at the other three KPIs. Similar situation happens to segment *Normal B*, where boots can be clearly seen in both *CPU LOAD* and *ETH INFLOW*. Therefore, we need to consider the full set of multivariate KPIs to pursue an accurate anomaly detection, as shown in segment *Anomaly A* and *Anomaly B*. Besides the dependency between KPIs, we can also leverage historical KPI patterns to reduce false positives. Specifically, in Figure 2.1, all KPIs have witnessed some abnormal spikes in history. However, they do not necessarily indicate the occurrence of failures. By learning such historical patterns, our model will not be too sensitive to spikes and will be able to distinguish benign change patterns from anomalous ones.

2.2 Problem Statement

Anomaly detection on system KPIs is crucial for proactive and prompt troubleshooting, aiming to discover abnormal status exhibited by the KPIs. Such anomalies often indicate that a system encounters faults or attacks. In this paper, we focus on anomaly detection for an entity based on the multivariate KPIs collected from it at equal-space timestamps [23]. The problem can be formally defined as follows.

The input of multivariate KPIs can be represented as $X \in \mathbb{R}^{n \times m}$, where n is the number of different KPIs and m is the number of observations. The t^{th} row of X, denoted as $x_t = [x_t^1, x_t^2, ..., x_t^m]$, is a m-dimensional vector containing the observation of each KPI at timestamp t. Similarly, the k^{th} column of X, denoted as $x^k = [x_1^k, x_2^k, ..., x_n^k]$, is a n-dimensional vector containing the observations of the k^{th} KPI. Particularly, we denote $x_{i:j}^k = [x_i^k, x_{i+1}^k, ..., x_j^k]$ as a consecutive sequence of x^k from timestamp i to j.

The objective of anomaly detection for multivariate KPIs is to determine whether or not a given x_t is anomalous, i.e., whether the entity is in abnormal status at timestamp t. For each timestamp t, our model calculates an anomaly score $s_t \in [0, 1]$, which represents the probability of x_t being anomalous. If s_t is larger than a pre-defined threshold θ, x_t will be predicted as an anomaly. The ground truth $\mathbf{y} \in \mathbb{R}^n$ is an n-dimensional vector consisting 0 and 1, where 0 indicates a normal point, and 1 indicates an anomalous one.

Chapter 3

Methodology

In this section, we present CMAnomaly, our framework of anomaly detection on multivariate KPIs. We first give an overview of CMAnomaly and then elaborate on the details.

3.1 Overview of CMAnomaly

In today's software systems, e.g., online service systems, engineers face an overwhelming number of KPIs, which are being generated in a 24×7 non-stop basis. It would be labor-intensive and error-prone to manually investigate each KPI for failure detection and diagnosis. CMAnomaly facilitates this process by providing automated and effective multivariate KPI anomaly detection.

The overall framework of CMAnomaly is illustrated in Figure 3.1, which consists of four phases, i.e., *data preprocessing*, *collaborative machine*, *model training*, and *anomaly detection*. The first phase preprocesses the data by applying normalization and window sliding. Particularly, the input types of KPIs can vary depending on the application scenario. These KPIs can be unified to the same range by normalization. In the next phase, the prepro-

cessed data are fed to the proposed collaborative machine, which is the core component of CMAnomaly. The collaborative machine can properly capture the interactions among multivariate KPIs along with both feature and temporal dimensions. In the third phase, we train a forecasting-based anomaly detection model [6, 12], which detects anomalies based on prediction errors. Thanks to the collaborative machine, we can employ a cost-effective model architecture. Both of the above two phases play a crucial role in achieving a highly accurate and efficient anomaly detector. Finally, the trained model will be applied to detect anomalies for new observations.

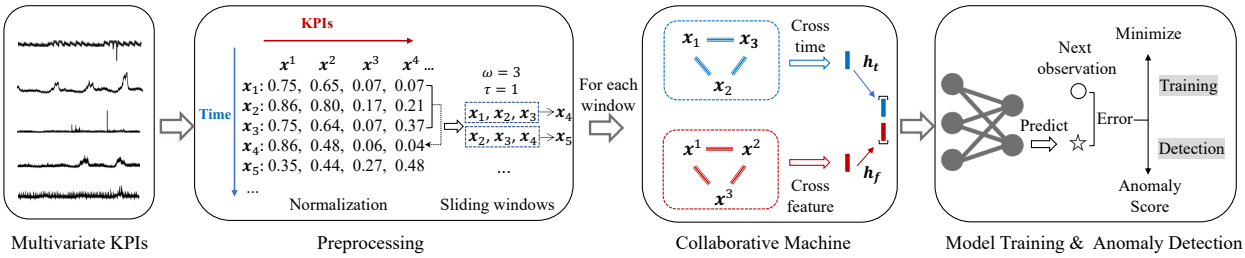

Figure 3.1: Overall Framework of CMAnomaly

3.2 Preprocessing

Different KPIs may have distinct scales, for example, the KPI monitoring the CPU execution, i.e., *CPU USAGE*, is in the range of 0% to 100%, However, the KPI monitoring the network traffic, i.e., *INBOUND PACKAGE RATE* can range from zero to millions of kilobytes. Therefore, data normalization is performed for each individual KPI to ensure the robustness of our model. After that, we conduct a sliding window to generate input to the model. Each window preserves the local pattern of multivariate KPIs.

3.2.1 Data Normalization

We apply max-min normalization to each individual KPI, i.e., x^k, as follows:

$$x^k_{norm} = \frac{x^k - \min(x^k)}{\max(x^k) - \min(x^k)}, \tag{3.1}$$

where the values of $max(x^k)$ and $min(x^k)$ are computed in the training data, which will then be used for test data normalization. For simplicity, we omit the "norm" subscript in the following elaboration.

3.2.2 Sliding Window

The sliding window is to partition KPIs along the temporal dimension. Particularly, it consists of two attributes, i.e., window size ω and stride τ. The stride indicates the forwarding distance of the window along the time axis to generate multivariate KPI windows. As the stride is often smaller than the window size, there exists overlapping between two consecutive windows. We denote the s^{th} sliding window as:

$$X_s = [x_{s\tau}, x_{s\tau+1}, ..., x_{s\tau+\omega-1}] \tag{3.2}$$

where $s \in [0, 1, 2, ...]$. X_s together with the observations at the next timestamp of the window, i.e., $\hat{x}_s = x_{s\tau+\omega}$, constitute a pair $(X_s, \hat{x}_s)$, where $X_s \in \mathbb{R}^{\omega \times m}$ and $\hat{x}_s \in \mathbb{R}^m$.

3.3 Collaborative Machine

As shown in Figure 2.1, to detect anomalies on multivariate KPIs accurately, we need to consider the dependency between KPIs. Moreover, historical

patterns of KPIs also provide important clues for anomaly detection. Specifically, if spikes often appear in the history and no failures are reported. By learning such benign change patterns, our model can alleviate the issue of noise in KPIs.

In this section, we introduce the proposed collaborative machine, a mechanism to efficiently capture the interactions of multivariate KPIs along with both the feature and temporal dimensions. Specifically, we allow cross-feature and cross-timestamp interactions by taking the inner-product of pairwise KPIs and pairwise timestamps in a multivariate KPI window as shown in Figure 3.1. The results are two types of vectors, i.e., KPI feature vector and temporal vector, which captures the dependencies and historical patterns of KPIs, respectively. To reduce the computation complexity, inspired by factorization machine [20], we reformulate each KPI inner-product and timestamp inner-product pair to achieve a linear time complexity. Finally, to simultaneously consider both types of features, we concatenate them and feed them to a cost-effective model for training.

3.3.1 Multivariate KPIs Interactions

To explicitly capture the dependency between multivariate KPIs and their historical patterns, for each sliding window, denoted as $X_s \in \mathbb{R}^{\omega \times m}$, we calculate the pairwise inner product of all KPI feature vectors, i.e., $x^k_{s\tau, s\tau+\omega-1}, k \in [1, m]$, and temporal vectors, i.e., $x_t, t \in [s\tau, s\tau + \omega - 1]$. Particularly, for notation simplification, we remove the subscript of feature vectors.

$$h_f = b_0 + \sum_{i=1}^{m} w_i x^i + \sum_{i=1}^{m} \sum_{j=i+1}^{m} \left\langle x^i, x^j \right\rangle v_i v_j \tag{3.3}$$

$$h_t = \hat{b}_0 + \sum_{i=1}^{\omega} \hat{w}_i x_i + \sum_{i=1}^{\omega} \sum_{j=i+1}^{\omega} \left\langle x_i, x_j \right\rangle \hat{v}_i \hat{v}_j \tag{3.4}$$

The cross-feature and cross-time KPI interactions, denoted as h_f and h_t, are formulated as Equation 3.3 and 3.4, respectively. Particularly, we only elaborate on the Equation 3.3 as Equation 3.4 shares the same format. In Equation 3.3, $b_0, w_i, v_j, v_j \in \mathbb{R}$ are trainable parameters, x^i, $x^j \in \mathbb{R}^\omega$ are the i^{th} and j^{th} column of X_s with each column representing all the observations of a KPI in the corresponding window, and $< \cdot, \cdot >$ is the operation of inner product. The equation is composed of three terms: the first term is a trainable bias, the second term is a weighted sum of all KPIs without explicit interaction, and the third term is the core part of the proposed collaborative machine, which models the pairwise KPI interactions. Specifically, each pair of KPI interactions is modeled by the inner product of the corresponding two KPIs. However, not all the pairs share the same importance. To model the importance of each KPI and its interaction with other KPIs, we assign a trainable weight to each particular KPI, e.g., v_i for the i^{th} KPI. After that, while interacting with other KPIs, the product of the weights can measure the importance of the combination, e.g, $v_i v_j$ denotes the importance of the interaction between the i^{th} and j^{th} KPIs.

3.3.2 Efficient Computation

As all pairwise feature and temporal interactions require to be computed, the operations shown in Equation 3.3 and 3.4 are computational expensive, i.e., $\mathcal{O}(m^2)$ and $\mathcal{O}(\omega^2)$, respectively. They are not practical for industrial scenarios

as a small increase in the number of multivariate KPIs or observation step size, i.e., larger m and ω, will quickly increase the computation complexity. To capture the feature and temporal interactions more efficiently, inspired by the work of Rendle et al. [20], we reformulate the pairwise interaction term in Equation 3.3 and 3.4, which could greatly reduce the computation cost, leading to a linear time complexity. Particularly, we only demonstrate the reformulation of the feature interaction term in Equation 3.3, as the temporal version in Equation 3.4 can be derived similarly:

$$
\begin{aligned}
&\sum_{i=1}^{m}\sum_{j=i+1}^{m}\left\langle x^i, x^j\right\rangle v_i v_j \\
=&\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\left\langle x^i, x^j\right\rangle v_i v_j - \frac{1}{2}\sum_{i=1}^{m}\left\langle x^i, x^j\right\rangle v_i v_i \\
=&\frac{1}{2}\left(\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{r=1}^{\omega} x_r^i x_r^j v_i v_j - \sum_{i=1}^{m}\sum_{r=1}^{\omega} x_r^i x_r^i v_i v_i\right) \\
=&\frac{1}{2}\sum_{r=1}^{\omega}\left(\left(\sum_{i=1}^{m} x_r^i v_i\right)\left(\sum_{j=1}^{m} x_r^j v_j\right) - \sum_{i=1}^{m}(x_r^i)^2 v_i^2\right) \\
=&\frac{1}{2}\sum_{r=1}^{\omega}\left(\left(\sum_{i=1}^{m} x_r^i v_i\right)^2 - \sum_{i=1}^{m}(x_r^i)^2 v_i^2\right)
\end{aligned}
\tag{3.5}
$$

Equation 3.5 is mathematically equivalent to the third term of Equation 3.3. However, the computation complexity is decreased from $\mathcal{O}(m^2)$ to $\mathcal{O}(m\omega)$. Particularly, considering all windows generated from the n observations, the overall computation is $\mathcal{O}(mn)$. Moreover, the computation involved in Equation 3.5 can be implemented as matrix multiplications, which can be accelerated by GPU (graphics processing unit) for better efficiency. By substituting the interaction terms in both Equation 3.3 and 3.4, we can

capture the pairwise cross-feature and cross-time KPI interactions to produce h_f and h_t in a highly efficient manner.

3.4 Model Training and Anomaly Detection

The last two phases of CMAnomaly are model training and anomaly detection. The anomaly detection model of CMAnomaly works in a forecasting manner. To be more specific, the model is trained to predict the next KPI values given preceding observations. Anomaly is then detected based on prediction errors, as shown in Figure 3.1. In the training phase, as most multivariate KPIs would reflect the normal status of an entity, the model will learn the normal patterns of KPIs, i.e., what the next observations would be given previous ones. Although there could be anomalies in the training data, they tend to be forgotten by the model as they rarely appear. Consequently, in the detection phase, the model will predict "normal" KPI values based on the learned patterns. If the real observations deviate from the predicted ones by a significant margin, an anomaly may happen, i.e., the entity is not in its normal status. Particularly, the deviation measures the likelihood of the occurrence of the anomaly.

Our framework supports various types of neural network models for anomaly detection. However, thanks to the formulation of KPI interactions in Equation 3.3 and 3.4, we find a cost-effective model architecture, i.e., multi-layer perceptron (MLP), is sufficient to achieve competitive performance. The anomaly detection model can be formulated as follows:

$$\tilde{h}_{i+1} = \sigma(\tilde{h}_i \tilde{X}_i + \tilde{b}_i), i = 0, 1, ..., L-1, \tag{3.6}$$

where L is the number of layers of the MLP model, $\tilde{W}_i, \tilde{b}_i$ are trainable pa-

rameters with customizable size, and $\sigma(x) = max(0, x)$ is the ReLU activation function. Particularly, we simultaneously consider the cross-feature and cross-time KPI interactions by concatenating h_f and h_t, which is the input to the model, i.e., $\tilde{h}_0 = concat(h_f, h_t)$. $\hat{y} = \tilde{h}_L \in \mathbb{R}^m$ is the prediction result produced by the last layer of the MLP model, which contains the predicted values for all KPIs at the next timestamp.

3.4.1 Training

We train the anomaly detection model by minimizing the following mean squre error (MSE) loss $\mathcal{L}$ between the predictions and ground truth observations:

$$\mathcal{L} = \sum_{i=1}^{N} \left\| \hat{y}_i - \hat{x}_i \right\|_2 , \tag{3.7}$$

where N is the number of training sliding windows. $\hat{y}_i \in \mathbb{R}^m$ and $\hat{x}_i = x_{i\tau+\omega} \in \mathbb{R}^m$ are the predicted and ground truth observations for the i^{th} window, respectively. The minimization of $\mathcal{L}$ is conducted using the Adam optimizer [13]. Particularly, we stop the training process when the loss $\mathcal{L}$ is relatively stable, i.e., the change of $\mathcal{L}$ is less than 10^{-5}. With the minimization of loss $\mathcal{L}$ during training, CMAnomaly can learn from the normal patterns in the training data by updating all trainable parameters, e.g., $v_i v_j$ denoting the interaction weights.

3.4.2 Detection

After the model is trained, we compute an anomaly score for each window X_i in the testing data. Specifically, we first calculate the MSE between the predicted and ground truth observations, and then apply the sigmoid function

to rescale the score to the range $[0, 1]$, which represents the probability of the occurrence of an anomaly:

$$s_i = \phi\left(\frac{1}{m}\left\|\hat{y}_i - \hat{x}_i\right\|_2\right) \tag{3.8}$$

where $\phi(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function.

To determine whether an anomaly has happened, a threshold θ should be defined for the anomaly score. The timestamps with a large anomaly score, i.e., $s_i \geq \theta$, should be regarded as anomalous points. In reality, the threshold can be set by on-site engineers based on their experience. A large threshold imposes a strict anomaly detection policy, which may miss important system failures, i.e., low recall. However, a small threshold increases the sensitivity to KPI changes, resulting in false alarms, i.e., low precision. In our experiments, we empirically set the threshold that yields the best experimental results.

Chapter 4

Experiments

In this section, we evaluate CMAnomaly using both public data and industrial data. Particularly, we aim to answer the following research questions (RQs):

RQ1: How effective is CMAnomaly in multivariate KPI anomaly detection compared with state-of-the-art approaches?

RQ2: Can the proposed collaborative machine mechanism facilitate anomaly detection?

RQ3: How efficient is CMAnomaly compared with existing approaches?

4.1 Experimental Setup

4.2 Dataset

To evaluate the effectiveness of CMAnomaly, we conduct experiments on three public datasets. Moreover, to confirm its practical significance, we collect an industrial dataset from the Networking service of Huawei Cloud.

Public dataset. The three public datasets used in our experiments are SMAP (Soil Moisture Active Passive satellite), MSL (Mars Science Laboratory rover), and SMD (Server Machine Dataset). SMAP and MSL are two

real-world datasets released by NASA. They are collected from the running spacecraft and contain a set of telemetry anomalies corresponding to actual spacecraft issues involving various subsystems and channel types [12]. SMD dataset is collected from a large Internet company containing a 5-week-long monitoring KPIs of 28 machines released by Su et al. [23].

Industrial dataset. Besides the public dataset, we also collected real-world KPIs from a global cloud service provider Huawei Cloud to conduct a more comprehensive evaluation. To support tens of millions of users worldwide, the network service of Huawei Cloud contains a large number of nodes. Therefore, to provide a stable 24×7 service, the status of each component of the network is closely monitored with KPIs. The engineers can fix problematic components timely if the anomalies of KPIs can be automatically detected and reported in real-time. To evaluate our method in a practical scenario, we collected a 30-day-long KPIs dataset with 13 network components within Jan. 2021. The dataset is termed as *Industry* in Table 4.1. Each of the network components has 70$\sim$200 different types of KPIs. We use the first 20 days of KPIs as the training data and the rest as the testing data. Then, several experienced engineers are invited to manually label the anomalous points in the testing data. In this process, disagreement on labels will be discussed until consensus is reached. We plan to open-source this dataset to facilitate future studies in this field[1].

The statistics of the datasets are listed in the Table 4.1. *#Train* and *#Test* denote the number of observations in the training and testing dataset, respectively. *#Entities* denotes the number of entities in a dataset. Specifically, in SMAP and MSL, the entity is the spacecraft. In SMD, the entity is a server machine. In our industrial dataset, an entity is a pod in the network service consisting of a number of routers and servers. *#Dim.* stands for the

[1]We will open-source them after double-blind review.

number of KPIs associated with an entity. *Anomaly Ratio* is the ratio of the number of anomalous points to the number of observations.

Table 4.1: Dataset Statistics

Dataset	#Train	#Test	#Entities	#Dim.	Anomaly Ratio (%)
SMAP	135,183	427,617	55	25	13.13
MSL	58,317	73,729	27	55	10.72
SMD	708,405	708,420	28	38	4.16
Industry	1,728,000	864,000	13	70∼200	1.16

4.3 Experimental Environment

We run all experiments on a Linux server with Intel Xeon Gold 6148 CPU @ 2.40GHZ and 1TB RAM, GeForce RTX 2080 Ti, running Red Hat 4.8.5 with Linux kernel 3.10.0. In addition, the proposed model is implemented under the PyTorch framework and runs on the GPU.

4.4 Evaluation Metrics

As anomaly detection is a binary classification problem, we employ Precision (PC), Recall (RC), and F1 score (F1) for evaluation, as calculated by Equation 4.1. Specifically, TP (true positive) denotes the number of anomalous samples that the model correctly predicts as an anomaly. FP (false positive) is the number of normal samples that the model mistakenly recognizes as an anomaly. FN (false negative) is the number of normal samples that are predicted as normal. F1 score is the harmonic mean between precision and recall. A higher F1 score indicates a better model. To show the best perfor-

mance of our method, we manually tune the anomaly threshold to achieve the best F1 score as done by [1].

$$PC = \frac{TP}{TP + FP} \quad RC = \frac{TP}{TP + FN} \quad F1 = 2 \cdot \frac{PC \cdot RC}{PC + RC} \tag{4.1}$$

In reality, failures often last for a certain period, which results in consecutive anomalies in a time series. Therefore, it is practical to consider that a model makes a correct prediction for an anomalous segment if at least one point in the segment is successfully predicted as an anomaly. We follow the point adjustment process as done in [1,19,23]. For example, as shown in Figure 4.1, seven points are predicted by the model, whose results are indicated as *Prediction*. For the 4^{th} prediction, an anomaly is reported, which is within the range of the anomaly of the ground truth (2^{nd} to 5^{th} points). In this case, we adjust the prediction to be *Adjusted Prediction* by converting zeros to ones in this range. Finally, we report the evaluation results computed with the ground truth and the adjusted prediction.

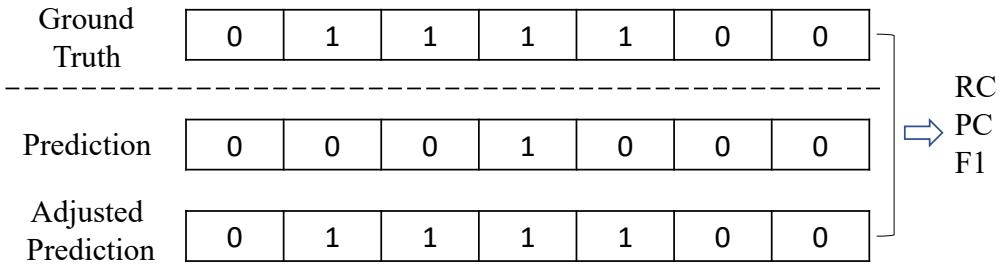

Figure 4.1: Point Adjustment Process

4.5 RQ1: Effectiveness of CMAnomaly

To study the effectiveness of CMAnomaly, we compare its performance with various state-of-the-art baseline models on both the public datasets and the

Table 4.2: Accuracy Comparison of Different Anomaly Detection Methods on Public Datasets

Methods	SMAP			MSL			SMD			Average		
	PC	RC	F1	PC	RC	F1	PC	RC	F1	PC	RC	F1
IF	0.4423	0.5105	0.4671	0.5681	0.6740	0.5984	0.5938	0.8532	0.5866	0.5347	0.6792	0.5507
LSTM-NDT	0.8965	0.8846	0.8905	0.5934	0.5374	0.5640	0.5684	0.6438	0.6037	0.6861	0.6886	0.6861
DAGMM	0.6334	0.9984	0.7124	0.7562	0.9803	0.8112	0.6730	0.8450	0.7231	0.6875	0.9412	0.7489
LSTM-VAE	0.7164	0.9875	0.7555	0.8599	0.9756	0.8537	0.8698	0.7879	0.8083	0.8154	0.9170	0.8058
AE	0.7216	0.9795	0.7776	0.8535	0.9748	0.8792	0.8825	0.8037	0.8280	0.8192	0.9193	0.8283
OmniAnomaly	0.7585	0.9756	0.8054	0.9140	0.8891	0.8952	0.9809	0.9438	<u>0.9441</u>	0.8845	0.9362	0.8816
USAD	0.7697	0.9831	0.8186	0.8810	0.9786	<u>0.9109</u>	0.9314	0.9617	0.9382	0.8607	0.9745	0.8892
MTAD-GAT	0.8906	0.9123	<u>0.9013</u>	0.8754	0.9440	0.9084	-	-	-	0.8830	0.9282	<u>0.9049</u>
CMAnomaly	0.9989	0.8916	**0.9187**	0.9903	0.9690	**0.9782**	0.9658	0.9379	**0.9512**	0.9850	0.9328	**0.9494**

industrial dataset collected from Huawei Cloud. We first train CMAnomaly on the training data until convergence. Then, we compute the anomaly score for each observation of the testing data. After that, we manually try thresholds θ in the range of $[0, 1]$ with a step size of 0.1 to produce the prediction. In particular, for the baselines with automatic threshold selection mechanism, we replace such mechanism with manual selection. In doing this, we follow a recent work [1] to obtain the best performance of all models for a fair comparison. Finally, the predictions of all models are adjusted to compute PC, RC, and F1.

Performance on public datasets. We select seven unsupervised multivariate KPI anomaly detection methods as the baselines, which are isolation forest (IF) [16], LSTM-NDT [12], DAGMM [32], LSTM-VAE [18], autoencoder (AE) [1], OmniAnomaly [23], USAD [1] and MTAD-GAT [29]. Particularly, we directly reuse the results reported by Audibert et al. [1] because (1) the set of datasets employed in their experiments is the same as ours, and (2) we follow their procedure of model tuning.

The overall performance is shown in Table 4.2, where we mark the best F1 score in bold and underline the second-best ones. We also report the average

metrics on all datasets in the "Average" columns. We have the following observations: (1) IF achieves the worst performance compared with other baseline models. Because IF tries to isolate the anomalous timestamp independently, which loses the temporal dependency. (2) Other LSTM-based or AE-based methods including LSTM-NDT, DAGMM, LSTM-VAE, AE perform much better than IF and achieve 0.6861 $\sim$ 0.8283 F1 score because these models take a window of observations as input, which helps to retain valuable historical information. (3) OmniAnomaly and USAD outperform other baselines by a large margin and achieve nearly 0.88 f1 score. These two methods introduce different mechanisms to ensure robust anomaly detection. OmniAnomaly models MTS through stochastic variables and uses reconstruction probabilities to determine anomalies. Differently, USAD utilizes the generative adversarial network (GAN) to train autoencoders, which allows the model to detect anomalies close to normal data.

(4) The proposed method CMAnomaly achieves the highest F1 score on all datasets. The average F1 score is significantly improved to 0.9494 from the second-best, i.e., 0.9049, achieved by MTAD-GAT. Both of CMAnomaly and MTAD-GAT explicitly model the dependency of KPIs. It shows that such explicit modeling is superior to implicit modeling of other baselines. However, the cost-effective design of CMAnomaly allows it to incur less trainable parameters than MTAG-GAT. As a consequence, overfitting to the training data could be alleviated. In other words, only normal patterns with relatively high occurrence are learned by CMAnomaly during training. When making anomaly detection on the test data, CMAnomaly can be less sensitive and only raise the anomaly score when a real anomaly happens, avoiding more false alarms than MTAD-GAT. The experimental results indicate that the PC of CMAnomaly is significantly higher than MTAD-GAT (0.9850 vs 0.8830), which implies fewer false alarms.

Table 4.3: Accuracy Comparison on Industrial Dataset

Methods	Industry		
	PC	RC	F1
OmniAnomaly	0.6639	0.8382	0.7283
LSTM-VAE	0.8273	0.7436	0.7560
CMAnomaly	0.9179	0.8202	**0.8368**

Performance on industrial dataset. To show the superiority of CMAnomaly in the industrial application, we apply two most effective open-source anomaly detection methods, i.e., LSTM-VAE [18] and OmniAnomaly [23] on the industrial dataset for comparison.

The experimental results are shown in Table 4.3. In particular, the PC of OmniAnomaly is the lowest, but the RC is the highest because the complex architecture of OmniAnomaly incurs more trainable parameters, which makes it easier to overfit the training data. Therefore, OmniAnomaly is more sensitive to capture more anomalies but has the most false positive alarms. Different from the results in Table 4.2, LSTM-VAE outperforms OmniAnomaly on this dataset in terms of F1. LSTM-VAE has a more light-weight design than OmniAnomaly, so LSTM-VAE suffers less overfitting. As a result, LSTM-VAE only raises the anomaly score when the new observation deviates more from the prediction. In this case, though higher PC is achieved, LSTM-VAE has the lowest RC because it cannot effectively find all possible anomalies. CMAnomaly can balance PC and RC better and achieves the best F1 score, $\sim$0.08 higher than the second-best one achieved by LSTM-VAE. The collaborative machine facilitates CMAnomaly to capture the dependency of the training KPIs effectively. Therefore, CMAnomaly avoids overfitting the noisy points existing in training data e.g., usual spikes as shown in Figure 2.1. Instead, CMAnomaly reports a higher anomaly score only when the dependent KPIs are anomalous, thus achieves the highest precision.

4.6 RQ2: Effectiveness of Collaborative Machine

To evaluate the effectiveness of the proposed collaborative machine used in CMAnomaly. We firstly produce a variant of CMAnomaly by removing the collaborative machine (CM). Specifically, we use the average of all the KPIs across the time dimension as the input of the MLP instead of hidden vectors computed by Equation 3.3 and Equation 3.4. In doing this, explicit interactions are removed. Then we apply the variant to perform anomaly detection on three public datasets SMAP, MSL and SMD. Then, we compare the F1 score obtained by the variant with CMAnomaly. Figure 4.2 shows the accuracy comparison between the variant and CMAnomaly. The variant is denoted as "CMAnomaly w/o CM" in the figure. We can observe that the accuracy of the variant drops on all three datasets. Especially, the average accuracy decreases by ~ 0.07 from 0.9494 to 0.8789. This can be explained by the loss of valuable temporal and feature information of the variant. On the one hand, without the collaborative machine, the sequential dependency and feature dependency cannot be captured, which hinders the model to learn from the normal patterns in the training data. As a consequence, when making the prediction in the anomaly detection stage, the new normal observation cannot be accurately predicted thus larger anomaly score may be produced. Therefore, more false alarms are reported.

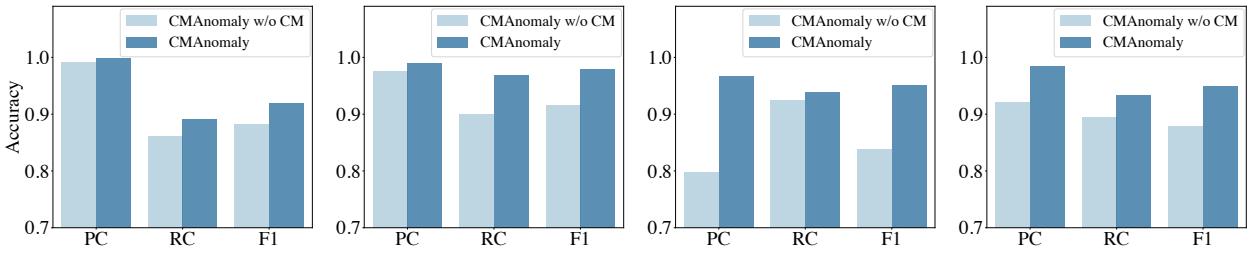

Figure 4.2: Performance Comparison with and without Collaborative Machine

4.7 RQ3: Efficiency of CMAnomaly

In this section, we evaluate the efficiency of CMAnomaly compared with the baseline models. We select the OmniAnomaly and LSTM-VAE for comparison because they are the most effective methods that are open-source. We conduct the comparison based on the official implementation of OmniAnomaly[2] and LSTM-VAE[3]. Next, we perform these methods on the industrial dataset and record training time and prediction time. Training time defines as the total time of feeding the preprocessed data to the model, and prediction is the total time used to predict the next observation on all the windows in the test set. Specifically, for a fair comparison, all the models are trained in a batch-wise manner for one epoch, and the same batch size is used. In real-world practice, we need to apply different window sizes to detect anomalies under different granularities. Therefore we increase the window size ω in the range of $[16, 32, 64, 128, 256]$ to evaluate the efficiency of different methods. Stride is set as $\tau = 5$ on the training data and $\tau = 1$ on the test data because anomaly detection on test data should be conducted under finer granularity to alleviate missing anomalies. All the experiments run on the same computation machine on a GPU of GeForce RTX 2080 Ti.

Figure 4.3 and 4.4 show the efficiency comparison of the selected models in the training and prediction phases, respectively. According to the results, we can see that (1) the computational cost of LSTM-VAE and OmniAnomaly increases quickly with a larger window size, because both of the methods adopt LSTM as a core part to handle sequential input. Therefore, when we increase the window size, the depth of the LSTM increases, thus incurs more computation. (2) OmniAnomaly costs more time (nearly $2\times$) than

[2]https://github.com/NetManAIOps/OmniAnomaly
[3]https://github.com/TimyadNyda/Variational-Lstm-Autoencoder

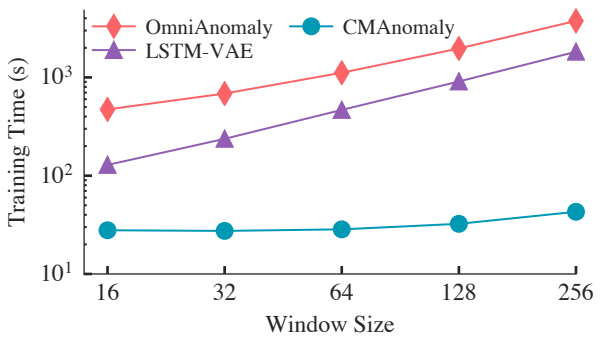

Figure 4.3: Training Time vs Window Size on Industrial Dataset

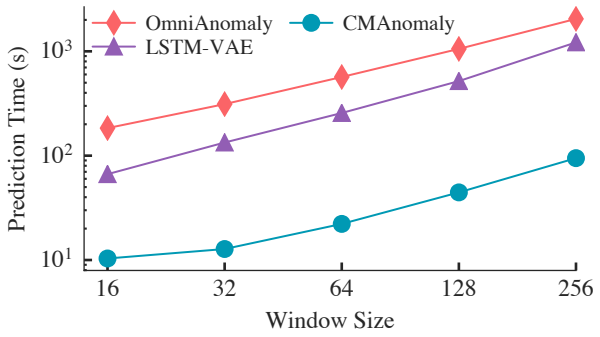

Figure 4.4: Prediction Time vs Window Size on Industrial Dataset

LSTM-VAE though they share similar LSTM and VAE architecture. Nevertheless, OmniAnomaly introduces additional stochastic variable connection and planar normalizing flow to model the KPIs, which are more computationally expensive. (3) CMAnomaly is the most efficient method. To be more specific, CMAnomaly is nearly 10x faster than LSTM-VAE and 20x than OmniAnomaly. Significantly, the training and prediction time do not increase a lot when varying the window size. Mainly, since τ on the test set is set to 1 and less than that of the training data, more test sliding windows are generated. Therefore, the test time of CMAnomaly is slightly larger than the training time. This only happens on CMAnomaly because the other two methods need more computation on the back-propagation during training.

The cost-effective design and reduction of the computation complexity of CMAnomaly benefit its efficiency. Moreover, both the collaborative machine and MLP of CMAnomaly can be well accelerated by the asynchronous computing architecture of GPUs. Therefore, CMAnomaly could conduct efficient training and prediction.

We want to emphasize that the efficiency of CMAnomaly can meet the demanding requirements of industrial scenarios. (1) Generally, a large-scale system is closely monitored by a large number of KPIs collected every second. Such a large quantity of KPIs cloud be processed by CMAnomaly in real-time. (2) CMAnomaly can be fast retrained to learn the new patterns of KPIs after the system upgrades. With this advantage, the deployment of an online anomaly detection service will not be suspended for too long.

4.8 Case Study

To make a more concrete evaluation of CMAnomaly, we provide a case study on how CMAnomaly computes anomaly scores for given multivariate KPIs. This case study is conducted on a segment of the industrial dataset. The KPIs monitor the traffic of connected network nodes supporting a service. After the model is trained, we apply the model to consecutively predict the KPIs and detect anomalies. The results are shown in Figure 4.5. In this figure, the first four rows show the real observation of the KPIs and the prediction made by CMAnomaly. The last row shows the anomaly score of the entity and a threshold selected. If we consider the KPIs individually, the first two KPIs have spikes at both segments marked as "Normal" and "Anomaly" so anomalies would be alarmed. Differently, our method computes a much lower anomaly score of the "Normal" segment than the "Anomaly" segment by simultaneously considering the dependency of all the KPIs. As a result,

the anomaly score at the true positive segment can be magnified. In fact, the "Normal" spikes are caused by a typical increase of user requests to a port of the service. However, in the "Anomaly" segment, a physical machine failure happens, which leads to the anomalous patterns of the KPIs.

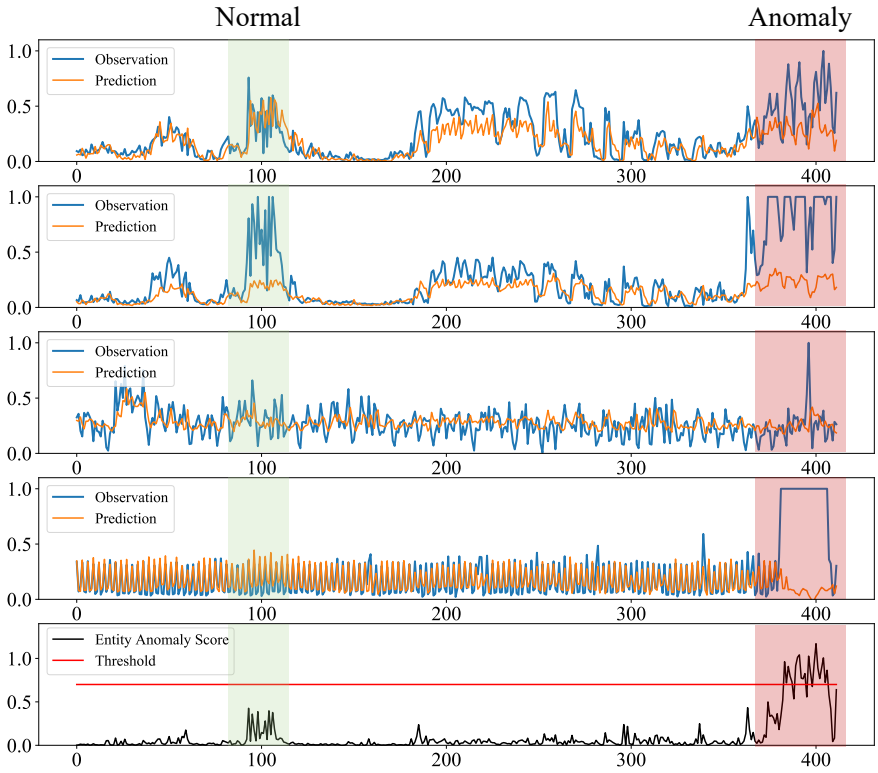

Figure 4.5: Case Study on Industrial Dataset

Chapter 5

Related Work

Anomaly detection on time series has been a hot topic and is widely studied. Key performance indicators (KPIs) used to monitor the runtime status of a system are one of the classic types of time series. In the literature, there are two categories of scenarios. One scenario needs to conduct anomaly detection for individual KPI, which can be solved by univariate time series models efficiently [19]. Another scenario needs to consider the dependency among multiple KPIs to detect the anomalous status of the unified unit.

Implicitly Modeling. The major challenge of multivariate anomaly detection comes from the effective modeling of complex temporal dependence and stochasticity of Multivariate KPIs. Much previous work attempts to capture the normal temporal dependence by modeling hidden states implicitly. For example, Hundman et al. [12] leverages LSTM without expert-labeled telemetry anomaly data to detect anomalies in multivariate time-series metrics of spacecraft based on prediction errors while maintaining interpretability throughout the system. Malhotra et al. [17] proposes an LSTM-based encoder-decoder network to reconstruct the "normal" time series with high probabilities. While the reconstruction errors are utilized to detect anomalies from multiple sensors. Another way to model normal patterns is to learn the

distribution of input data like deep generative models [7] and deep Bayesian network [24]. Donut [25] employs Variational AutoEncoder (VAE) to generate the normal hidden state of seasonal KPIs without expert-labeled data. With solid theoretical explanation, Donut successfully detects anomalies in seasonal KPIs with various patterns and data quality, but enjoys high time complexity in the training phase. To reduce the training complexity, DAGMM [32] simplify the hidden state as a combination of several Gaussian distributions. USAD [1] improves the autoencoder framework by incorporating adversarial samples to speed up the training phase. OmniAnomaly [23] proposes a stochastic recurrent neural network to captures the normal patterns of multivariate time-series by simulating normal data distribution through stochastic latent variables. Then, input data is reconstructed from the normal representation. Low reconstruction probability indicates anomalous input data. Similar to Hundman et al. [12]'s approach, OmniAnomaly provides interpretations based on the reconstruction probabilities of its constituent univariate KPI. However, the generalization capability of these generative approaches with implicitly modeling is degraded when they encounter severe noise in temporal KPIs, which is very common in industrial production systems.

Explicitly Modeling. To reduce the negative effect of noisy data in multivariate KPIs, the temporal dependencies such as the inter-correlations between different parts of KPIs should be captured in an explicit way. Zhang et al. [27] proposes a convolutional recurrent encoder-decoder framework called MSCRED to characterize multiple levels of the normal states in different steps of KPIs. MSCRED employs a convolutional encoder and convolutional LSTM network to capture the KPI interactions and temporal patterns, respectively. Finally, it generates different levels of anomaly scores based upon the severity of different incidents. Similar to our approach, Zhao et al. [29] consider each univariate KPI as an individual feature to capture the complex depen-

dencies of multivariate KPIs from temporal and feature perspective. Their proposed method incorporates feature-oriented and time-oriented graph attention mechanisms to obtain mixed hidden representation from a combination of forecasting-based and reconstruction-based frameworks. However, the overall framework enjoys high time-complexity and is hard to address the demanding industrial requirements for quick response and adjustment.

Chapter 6

Discussions

6.1 Success Story

In Huawei Cloud, CMAnomly has been successfully incorporated into the troubleshooting system, serving several essential services that have a large number of customers worldwide. We conduct field interviews with on-site engineers to collect feedback. Based on the feedback, we have seen it shedding lights on highly accurate and efficient anomaly detection on multivariate KPIs. Due to the complex system architecture, in Huawei Cloud, the number of KPIs being monitored is extremely large. Existing anomaly detection algorithms either report too many false positives or cannot meet the efficiency requirements, i.e., real-time detection and fast model retraining. Owing to the mechanism of collaborative machine and cost-effective model architecture, CMAnomly can achieve superior performance.

6.2 Lessons Learned

Utilizing expert knowledge. On-site engineers often possess decent knowledge about the systems. Such expert knowledge can facilitate model design

and development. For example, given thousands of KPIs at hand, they can quickly select the important ones that best characterize systems' health status. Moreover, instead of finding dependent KPIs from scratch as done in this paper, engineers can help identify a small set of correlated KPIs first. A more accurate model can then be pursued on top of these KPIs. However, such valuable knowledge is often not well accumulated, organized, and documented. For incident management in cloud systems, automated knowledge extraction [22] has been proposed. Efforts should also be devoted to system troubleshooting based on KPIs.

Building data collection pipeline. For modern software systems, IT operations play a crucial role for system reliability assurance [3, 5, 14, 30, 31]. Since it is data-driven by nature, sufficient and high-quality data are the foundation for model development in many applications, e.g., anomaly detection, failure diagnosis, fault localization. However, common data quality issues include insufficient labels, data noise, etc. To alleviate this issue, a complete and efficient pipeline should be constructed for monitoring data (e.g., KPIs and logs [9, 15, 26]) collection and storage. Specifically, when failure is detected, there should be tools to facilitate the labeling work for engineers. The labeled data should also be properly stored in a database which supports easy query in future. Once there are sufficient labels, possible supervised learning models can be explored.

Chapter 7

Conclusion

In this paper, we propose CMAnomaly, an anomaly detection framework for multivariate KPIs based on collaborative machine. Particularly, the proposed collaborative machine can learn the pairwise cross-feature and cross-time interactions between KPIs with linear time complexity. Thus, CMAnomaly can quickly obtain a big picture of a system's health status for anomaly detection. We have conducted experiments with three public datasets and one industrial dataset collected from a large online service of Huawei Cloud. For public datasets, CMAnomaly has achieved an average F1 score of 0.9494, outperforming the existing best approach by a significant margin of 4.5%. Similarly, a 3.3% of accuracy gain is achieved in the industrial dataset. More importantly, for model training and prediction, CMAnomaly runs up to 20x faster than the baseline methods, demonstrating that CMAnomaly is capable of meeting the demanding industrial requirements in terms of effectiveness and efficiency. Our framework has been successfully incorporated into the troubleshooting system of Huawei Cloud. Feedback from on-site engineers confirms its practical usefulness. We also open source our tool and sample experimental data publicly on GitHub to benefit the community[1].

[1]Due to the double-blind policy, the Github address will be released after paper review.

For future work, we will extend our framework by incorporating an adaptive anomaly detection mechanism based on human knowledge, i.e., human in the loop. Our model can efficiently adjust its decision behaviors with as fewer human efforts as possible.

Bibliography

[1] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga. USAD: unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th SIGKDD Conference on Knowledge Discovery and Data Mining, (KDD)*, pages 3395–3404. ACM, 2020.

[2] M. Braei and S. Wagner. Anomaly detection in univariate time-series: A survey on the state-of-the-art. *CoRR*, abs/2004.00433, 2020.

[3] J. Chen, X. He, Q. Lin, Y. Xu, H. Zhang, D. Hao, F. Gao, Z. Xu, Y. Dang, and D. Zhang. An empirical investigation of incident triage for online service systems. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, (ICSE-SEIP)*, pages 111–120, 2019.

[4] Z. Chen, Y. Kang, L. Li, X. Zhang, H. Zhang, H. Xu, Y. Zhou, L. Yang, J. Sun, Z. Xu, Y. Dang, F. Gao, P. Zhao, B. Qiao, Q. Lin, D. Zhang, and M. R. Lyu. Towards intelligent incident management: why we need it and how we make it. In *Proceedings of the 28th Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, (ESEC/FSE)*, pages 1487–1497, 2020.

[5] Y. Dang, Q. Lin, and P. Huang. Aiops: real-world challenges and research innovations. In *Proceedings of the 41st International Conference*

on Software Engineering: Companion Proceedings, (ICSE), pages 4–5, 2019.

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

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of the 27th Conference on Neural Information Processing Systems 2014, (NeurIPS)*, pages 2672–2680, 2014.

[8] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han. Outlier detection for temporal data: A survey. *IEEE Trans. Knowl. Data Eng.*, 26(9):2250–2267, 2014.

[9] P. He, Z. Chen, S. He, and M. R. Lyu. Characterizing the natural language descriptions in software logging statements. In *Proceedings of the 33rd International Conference on Automated Software Engineering, (ASE)*, pages 178–189, 2018.

[10] S. He, P. He, Z. Chen, T. Yang, Y. Su, and M. R. Lyu. A survey on automated log analysis for reliability engineering. *CoRR*, abs/2009.07237, 2020.

[11] S. He, Q. Lin, J. Lou, H. Zhang, M. R. Lyu, and D. Zhang. Identifying impactful service system problems via log analysis. In *Proceedings of the Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, (ESEC/FSE) FSE*, pages 60–70. ACM, 2018.

[12] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Söderström. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th International Conference on Knowledge Discovery & Data Mining, (KDD)*, pages 387–395, 2018.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of 3rd International Conference on Learning Representations, (ICLR)*, 2015.

[14] Q. Lin, K. Hsieh, Y. Dang, H. Zhang, K. Sui, Y. Xu, J. Lou, C. Li, Y. Wu, R. Yao, M. Chintalapati, and D. Zhang. Predicting node failure in cloud service systems. In *Proceedings of the Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, (ESEC/FSE)*, pages 480–490, 2018.

[15] Q. Lin, H. Zhang, J. Lou, Y. Zhang, and X. Chen. Log clustering based problem identification for online service systems. In *Proceedings of the 38th International Conference on Software Engineering, (ICSE)*, pages 102–111, 2016.

[16] F. T. Liu, K. M. Ting, and Z. Zhou. Isolation forest. In *Proceedings of the 8th International Conference on Data Mining (ICDM)*, pages 413–422, 2008.

[17] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection. *CoRR*, abs/1607.00148, 2016.

[18] D. Park, Y. Hoshi, and C. C. Kemp. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *CoRR*, abs/1711.00614, 2017.

[19] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th International Conference on Knowledge Discovery & Data Mining, (KDD)*, pages 3009–3017, 2019.

[20] S. Rendle. Factorization machines. In *Proceedings of the 10th International Conference on Data Mining, (ICDM)*, pages 995–1000, 2010.

[21] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.

[22] M. Shetty, C. Bansal, S. Kumar, N. Rao, N. Nagappan, and T. Zimmermann. Neural knowledge extraction from cloud service incidents. *CoRR*, abs/2007.05505, 2020.

[23] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th International Conference on Knowledge Discovery & Data Mining, (KDD)*, pages 2828–2837, 2019.

[24] H. Wang and D. Yeung. Towards bayesian deep learning: A framework and some existing methods. *IEEE Trans. Knowl. Data Eng.*, 28(12):3395–3408, 2016.

[25] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications.

In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, (WWW)*, pages 187–196. ACM, 2018.

[26] W. Xu, L. Huang, A. Fox, D. A. Patterson, and M. I. Jordan. Detecting large-scale system problems by mining console logs. In *Proceedings of the 22nd Symposium on Operating Systems Principles, (SOSP)*, pages 117–132, 2009.

[27] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *Proceedings of the 33rd Applications of Artificial Intelligence Conference, (AAAI)*, pages 1409–1416, 2019.

[28] X. Zhang, Q. Lin, Y. Xu, S. Qin, H. Zhang, B. Qiao, Y. Dang, X. Yang, Q. Cheng, M. Chintalapati, Y. Wu, K. Hsieh, K. Sui, X. Meng, Y. Xu, W. Zhang, F. Shen, and D. Zhang. Cross-dataset time series anomaly detection for cloud systems. In *Proceedings of the Annual Technical Conference, (USENIX ATC)*, pages 1063–1076, 2019.

[29] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang. Multivariate time-series anomaly detection via graph attention network. In *Proceedings of the 20th International Conference on Data Mining, (ICDM)*, pages 841–850, 2020.

[30] N. Zhao, J. Chen, X. Peng, H. Wang, X. Wu, Y. Zhang, Z. Chen, X. Zheng, X. Nie, G. Wang, Y. Wu, F. Zhou, W. Zhang, K. Sui, and D. Pei. Understanding and handling alert storm for online service systems. In *Proceedings of the 42nd International Conference on Software Engineering, Software Engineering in Practice, (ICSE-SEIP)*, pages 162–171, 2020.

[31] N. Zhao, J. Chen, Z. Wang, X. Peng, G. Wang, Y. Wu, F. Zhou, Z. Feng, X. Nie, W. Zhang, K. Sui, and D. Pei. Real-time incident prediction for online service systems. In *Proceedings of the 28th Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, (ESEC/FSE)*, pages 315–326. ACM, 2020.

[32] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *Proceedings of the 6th International Conference on Learning Representations, (ICLR)*, 2018.