Situation-Aware Multivariate Time Series Anomaly Detection Through Active Learning and Contrast VAE-Based Models in Large Distributed Systems

Zhihan Li, *Student Member, IEEE*, Youjian Zhao, Yitong Geng, Zhanxiang Zhao, Hanzhang Wang, Wenxiao Chen, Huai Jiang, Amber Vaidya, Liangfei Su, and Dan Pei, *Senior Member, IEEE*

***Abstract*— The massive amounts of monitoring data in network applications bring an urgent need for intelligent operation in large distributed systems. The key problem is precisely detecting anomalies in multivariate time series (MTS) monitoring metrics with the awareness of different application scenarios. Unsupervised MTS anomaly detection methods aim at detecting data anomalies from historical MTS without considering the out-of-band information (including user feedback and background information like code deployment status), which leads to poor performance in practice. To take advantage of the out-of-band information, we propose ACVAE, an MTS anomaly detection algorithm through active learning and contrast VAE-based detection models, which simultaneously learns MTS data's normal and anomalous patterns for anomaly detection. We also use a learnable prior to capture system status from the background information. Moreover, we propose a query model for VAE-based methods, which can learn to query labels of the most useful instances to train the detection model. We evaluate our algorithm on three different monitoring situations in eBay's search back-end systems. *ACVAE* achieves a range F1 score of 0.68∼0.96 with only 3% labels, significantly outperforming the best competing methods by 0.18∼0.50, and even better than a supervised ensemble method designed by domain experts in eBay.**

***Index Terms*— Anomaly detection, out-of-band information, multivariate time series, active learning, variational autoencoder.**

I. Introduction

THE rapid growth of network applications, *e.g.*, wearable devices, Internet of things (IoT), and Internet services, are all large-scale distributed systems. These systems generate massive amounts of monitoring data and bring an urgent need for intelligent network management [1]. Therefore, many research works are conducted on the intelligent operation [2]–[9] for monitoring these large-scale distributed systems, especially with recent popular deep learning [10] methods.

In this paper, we focus on large-scale Internet services (*e.g.*, search engines, e-commerce, social networks), whose backends are often large-scale distributed systems distributed in different data centers in the world. To ensure service stability and persistence, the key aspect of intelligent network operation is precise and **situation-aware** anomaly detection in different monitoring scenarios. Take the search backend system at eBay as an example. More than 300 million searches per day are done through eBay websites and mobile apps, serving 154 million users worldwide. The system is composed of thousands of microservice [11] applications with complex dependencies. As the main entrance of shopping orders in the company, it's important to maintain the system's stability. Anomalies such as significant search delays and app errors can have large impacts on the service quality and user experience, and a severe failure in the system can cause hundreds of thousands of dollars in financial loss per minute. Therefore, it's important to utilize advanced deep learning techniques to detect potential anomalies in time and maintain high service quality in different scenarios.

Specifically, several monitoring metrics (each of which is a univariate time series data, including CPU utilization, query per second, latency, *etc*.) are used to characterize the status of a system in different situations (*e.g.*, cluster machines, data centers with load-balance, deployment systems). These metrics compose the multivariate time series (MTS for short hereafter) data, which are often used as the input of anomaly detection algorithms to monitor whether there are potential failures in the whole system.

In recent years, several unsupervised deep learning based MTS anomaly detection algorithms have been proposed [2], [3], [12]–[17], which mainly try to learn the normal patterns of MTS from the historical raw MTS data to do anomaly detection. These methods have a common underlying assumption, *i.e.*, the "normal patterns" of data are generated from a deterministic procedure, and it's able to learn the distribution of "normal patterns" from the raw MTS data. Although these methods achieve high anomaly detection performance on some systems with *stable* services and *periodical* monitoring data,

Manuscript received 15 December 2021; revised 13 May 2022; accepted 20 June 2022. Date of publication 29 July 2022; date of current version 19 August 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB1802504 and in part by the State Key Program of National Natural Science of China under Grant 62072264. *(Corresponding author: Dan Pei.)*

Zhihan Li, Youjian Zhao, Wenxiao Chen, and Dan Pei are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: lizhihan17@mails.tsinghua.edu.cn; zhaoyoujian@tsinghua.edu.cn; chen-wx17@mails.tsinghua.edu.cn; peidan@tsinghua.edu.cn).

Yitong Geng, Zhanxiang Zhao, Hanzhang Wang, Huai Jiang, Amber Vaidya, and Liangfei Su are with eBay Inc., San Jose, CA 95125 USA (e-mail: yigeng@ebay.com; zhanzhao@ebay.com; hanzwang@ebay.com; huajiang@ebay.com; amvaidya@ebay.com; liasu@ebay.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2022.3191341.

Digital Object Identifier 10.1109/JSAC.2022.3191341

their assumptions may not be followed on many large-scale distributed systems with frequent changes and user-specific anomaly detection criteria, which leads to unsatisfactory detection performance.

Situation awareness is a key aspect of MTS anomaly detection in large distributed systems. In practice, the system operators detect anomalies not only based on raw MTS monitoring data, but also take other **out-of-band information** (including *background information* and *feedback information*) into consideration. For example, the monitoring system at eBay monitors millions of time-series data, including business (order volume, payment volume, product clicks, *etc*.) and operational metrics (throughput, CPU usage, memory usage, *etc*.). The definition of "normal patterns" for each metric varies according to different business implications so that the system operators have different needs for the anomaly detection system in different situations. Therefore, it's important to incorporate the out-of-band information into the deep learning algorithm for situation-aware MTS anomaly detection.

On the one hand, operators may pay attention to some **background information** in different systems when detecting anomalies, like time, code deployment status, promotion status, *etc*. For example, operators will treat a machine with statistically normal MTS patterns as an anomaly, if the machine should be under code deployment according to the recorded system status. Operators will also treat a rare volume spike in some business metrics as a normal event if they find it happens during a promotional period. These background information can be extracted from system logs or profiles, and encoded as one-hot vectors or real-valued series. Combining the background information and raw MTS data can help the detection algorithm be aware of the situations like system status, which improves the anomaly detection performance. On the other hand, operators' domain knowledge and business users' feedback can affect the anomaly detection criteria in different situations. These criteria can be summarised into **feedback information** (*i.e.*, labels) to guide the anomaly detection algorithm. For example, for the order volume metric, business people are more concerned about why the order decreased, but they are not concerned about the metric increase. Operators often do not care about the CPU or memory usage metrics decrease, and treat them as normal events. Operators do not care about the CPU usage spike during the batch job execution period for applications that have batch jobs every hour or day. The feedback information can reflect operators' anomaly detection criteria (*i.e.*, domain knowledge and experience) in different situations, which help the detection algorithm learn the operators' interests about anomalies in specific situations. Therefore, incorporating the out-of-band information can help the anomaly detection algorithm learn the operators' anomaly detection criteria and become situation-aware in different scenarios. This is extremely helpful in industrial systems, where the existing algorithms suffer from detecting statistically abnormal "data anomalies", but inconsistent with the operators' criteria in different situations (which are actually false positive alerts for operators).

It is also worth mentioning that, in practice, obtaining feedback information often needs discussion between system operators and business users. Therefore, only a little feedback can be provided in a low frequency. Thus, the anomaly detection algorithm needs to select the most useful instances (which help improve the detection model) to **query feedback**, which can proactively and precisely collect important feedback to improve the detection performance.

Therefore, there are *two main challenges* to incorporating out-of-band information (including feedback information and background information) for situation-aware MTS anomaly detection in large distributed systems. First, how to utilize the out-of-band information to help the model learn domain experts' criteria about anomalies and improve anomaly detection performance. Second, how to query the most useful instances and obtain user feedback, so as to improve the anomaly detection model with few labels.

In this paper, we propose *ACVAE*, a situation-aware MTS anomaly detection algorithm that incorporates out-of-band information through **A**ctive learning and **C**ontrast **V**ariational **A**uto**E**ncoder [18] (VAE) based models. To address the first challenge, *ACVAE* is composed of a conditional VAE-based structure to utilize the background information (*e.g.*, time, system flags) with a learnable prior, as well as a pair of contrast detection models ($\mathrm{model}_{\mathrm{full}}$ and $\mathrm{model}_{\mathrm{ano}}$) to take advantage of user feedback and simultaneously learn the normal and anomalous patterns of MTS in an active learning fashion. We also leverage a non-linear state space model structure in both $\mathrm{model}_{\mathrm{full}}$ and $\mathrm{model}_{\mathrm{ano}}$ to capture the temporal and correlational information inside MTS data. In this way, *ACVAE* continuously learns the difference between normal and anomalous patterns from the unlabeled data and labeled user feedback, and uses the difference of anomaly scores between $\mathrm{model}_{\mathrm{ano}}$ and $\mathrm{model}_{\mathrm{full}}$ as an effective anomaly indicator.

To address the second challenge, we propose a query model for VAE-based anomaly detection methods. Intuitively, anomalies are often rarer and more informative in anomaly detection, while the False Positives and False Negatives (FPs and FNs) in the detection results are also valuable for improving the detection model. Therefore, our query model takes the current detection models' outputs (including anomaly scores and standard deviation of posteriors) and historical queried labels as its input, aiming at finding the ground-truth anomalies and potential FPs and FNs in current detection results.

Through periodically training of the detection model and query model, *ACVAE* is able to improve the anomaly detection model as well as learn to query more useful instances. We evaluate the performance of *ACVAE* on MTS data collected from three different types of monitoring situations in search backend systems at eBay. The experimental results in Section IV-E show that *ACVAE* not only outperforms the state-of-the-art unsupervised and semi-supervised anomaly detection algorithms, but also beats the supervised ensemble method deployed in the production system of eBay.

The contributions of this paper are summarized as follows:

- To the best of our knowledge, we are the first to propose an end-to-end algorithm *ACVAE* that incorporates the out-of-band information (both background and feedback information) for situation-aware MTS anomaly detection in large distributed systems.

- For the first time, we propose a pair of contrast VAE-based models to model the normal and abnormal patterns for active MTS anomaly detection. We use two main designs in *ACVAE*, conditional VAE with learnable prior and contrast detection models, to tackle the challenge of utilizing out-of-band information for learning domain experts' criteria about anomalies in MTS anomaly detection.
- We propose a novel query model for VAE-based anomaly detection methods, which is able to proactively select the most useful instances for improving the detection model to ask for user feedback, so as to train the detection model through active learning with fewer feedback labels.
- We evaluate *ACVAE* on MTS data collected from three different monitoring scenarios in search backend systems at eBay. *ACVAE* achieves the best range F1-score from 0.68 to 0.96 with only 3% labels, outperforming the best unsupervised and semi-supervised baselines by 0.18 to 0.50 (23% to 289%). Moreover, *ACVAE* even achieves performance comparable to or better than the supervised ensemble method deployed in eBay, which demonstrates the feasibility of deploying *ACVAE* in production. We do case studies in production to show how *ACVAE* can reduce the number of False Positives and False Negatives with the help of out-of-band information. Furthermore, our ablation studies validate the effectiveness of each design in *ACVAE*.

II. Preliminaries

A. Multivariate Time Series Anomaly Detection and Out-of-Band Information

Multivariate Time Series (MTS) consists of a group of metrics, where each metric is a univariate time series, contains successive observations with equal-spaced sampling. Formally, the MTS data $\mathbf{x}_{1:n}^{1:m} \in R^{M \times N}$, where $n \in \{1, 2, \ldots, N\}$ are indexes of successive chronological observations, $m \in \{1, 2, \ldots, M\}$ are metrics. $\mathbf{x}_n^m$ is the value of metric m at index n.

In an industrial system, the MTS data is composed of several key performance metrics (*e.g.*, QPS, latency), which are used to monitor the health status of the whole system. The anomaly detection on MTS data focuses on detecting anomalous events in time, which can alert the corresponding operators to troubleshoot the problem.

Recent years, many unsupervised MTS anomaly detection algorithms [2], [3], [5], [12], [13], [15], [19] have been proposed, to learn MTS normal patterns (temporal or correlational information) from historical data. However, system operators detect anomalies not only relying on the raw MTS data, but also on other out-of-band information, including external events, time, domain knowledge, *etc*. Moreover, these criteria for anomalies often vary for different application scenarios. Therefore, it's hard for an unsupervised anomaly detection method to meet the demand of different industrial systems, since the system operators' interests about anomalies may not be completely embodied in the raw MTS data. In this way, *the anomaly detection algorithm needs to utilize the out-of-band information to learn the system operators' criteria about anomalies*, to reduce the number of false positive and false negative alerts and improve the anomaly detection performance.

B. Motivating Examples

In this section, we use several common cases in eBay's monitoring systems to show the necessity of learning operators' interests of anomalies with out-of-band information for practical situation-aware MTS anomaly detection.

Services usually perform regular jobs, such as backing up the database regularly, daily ETL tasks, *etc*. During the execution of the regular tasks, some metrics in the MTS will have changes. Fig. 1(a) shows part of a server machine's operational MTS. The server will get many requests from upstream applications at 0 o'clock on Monday every week to perform some timed tasks. We can see that the QPS and CPU usage (shown by the green line) will increase significantly during the execution of the task (the purple strip). Moreover, the period and duration of these tasks can vary in systems [20]. For many unsupervised methods [2], [3], [5], [14], [15], such spikes are rare features and are often falsely detected as anomalies without out-of-band information. However, this is a normal event for the operators in this scenario.

On the contrary, Fig. 1(b) is a part of the MTS of another server. This server will pull data from the upstream storage at 0 o'clock every day to calculate the amount of data and then update the cache. During the execution of the task, the CPU usage rate also soared. However, there was a problem with the upstream storage on the sixth day (shown by the red strip), and the data could not be pulled for subsequent calculations. The CPU usage did not have an obvious spike at 0 o'clock on that day. For system operators, this is an abnormal event. However, such smooth features (similar to its normal patterns) are often considered as a normal event for many unsupervised deep learning algorithms [2], [5], [13], [21] without considering out-of-band information.

In some clusters, when the operators or automated systems operate the cluster, a flag series is recorded to indicate the events on the cluster. The bottom three lines in Fig. 1(c) shows an example of flag series. Flag series record 0 most of the time, which means no special event happens. Flag series records 1 when a code change is made on this cluster and records 2 when the cluster is restarting. When events (*e.g.*, code deployment) happen on some clusters, their monitoring metrics (*e.g.*, CPU utilization) will have different patterns than usual, which are often falsely detected as anomalies by unsupervised methods, without considering the out-of-band flag information. For system operators, the anomaly criteria are to see if the clusters with the same flag (*i.e.*, same system status) follow similar correlations as usual, no matter how their metrics change due to the operation events. Series from top to bottom in Fig. 1(c) are CPU usage for group 0, 1, 2 (*i.e.*, MTS metrics) and their corresponding flag series 0, 1, 2 (*i.e.*, background information). According to the flag series, the operators only want to deploy code on groups 0 and 2. However, as shown in the red strip, group 1 should NOT be in deployment status, but its CPU

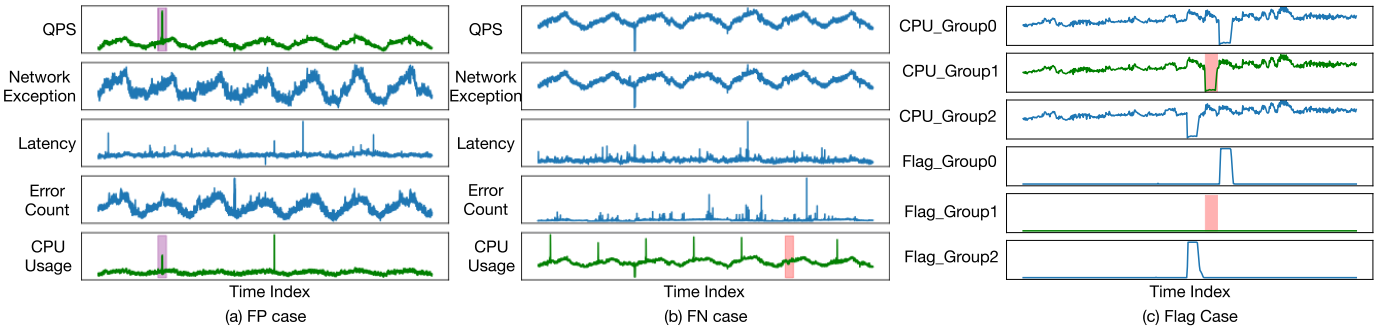


Fig. 1. Motivating examples for incorporating out-of-band information to improve MTS anomaly detection. The operators are more concerned about the metrics in the green line in these cases. (a) False Positive case, (b) False Negative case, (c) Flag case with background information.

metric shows a similar correlation with other groups that are in deployment. This suggests that there is an anomalous event in the deployment system. Therefore, the anomaly detection algorithm needs to take background information and user feedback into consideration to learn the operators' anomaly criteria in different scenarios.

C. Problem Statement

The examples in Section II-B motivate us to design semi-supervised anomaly detection methods that can better utilize the out-of-band information to improve MTS anomaly detection performance. Moreover, to achieve high anomaly detection performance with little feedback information, we mainly focus on active learning [22], [23], where the query model can proactively select the most useful instances for improving the detection model to query user feedback.

We formally define the MTS anomaly detection problem with out-of-band information as follows:

Input: raw MTS monitoring metrics $\mathbf{X}$, background information $\mathbf{T}$, partial user feedback (label) $\mathbf{Y}$.

Output: Whether a data point $x \in X$ is anomalous or not.

Specifically, the *raw MTS data* $\mathbf{X}$ is directly collected by the monitoring system, which may include payment volume, product clicks, CPU usage, memory usage, QPS, etc.

The *background information* $\mathbf{T}$ are extracted from structured system logs, which are also considered by the system operators to judge if an MTS observation is anomalous or not. The background information can be encoded as one-hot vectors (*e.g.*, timestamp in Section II-B case (a) and (b), indicators of different kinds of holidays, *etc*.) or real-valued series (*e.g.*, indicators of system status in Section II-B case (c), external events like $\{0, 1\}$ series reflecting whether a promotion is ongoing, *etc*.), which have the same length as raw MTS data $\mathbf{X}$. Although operators cannot directly *determine* whether an observation is anomalous by the background information, the background information is essential for human operators to be aware of the situations (*e.g.*, system status) and detect anomalies. Therefore, we use background information $\mathbf{T}$ as part of the input of our algorithm, which helps it learn human operators' anomaly detection criteria in different scenarios.

Finally, the *feedback information* $\mathbf{Y}$ are provided by the system operators, which directly reflect the operators' judgment for a given observation. For simplicity, here we only use labels (*i.e.*, $y_i \in \{0, 1\}, y_i \in \mathbf{Y}$) as feedback, where $y_i = 0$ means normal observation and $y_i = 1$ means anomalies. With the feedback label, human operators can feed their criteria of normal and anomalies for a specific system to the anomaly detection algorithm. In order to provide the feedback, the operators may consider more domain knowledge that cannot be encoded in the background information (*e.g.*, the criteria to distinguish anomaly and normal fluctuations, the special rule for current system, etc) and discuss with the downstream users to learn the events' effects. Leveraging the feedback information can help the anomaly detection algorithm learn the operators' interests of anomalies in different situations, so as to give more precise and situation-aware detection results.

In general, the status of a system can be captured by the joint distribution $P(\mathbf{X}, \mathbf{T})$ (*i.e.*, the raw MTS metrics and background information), as well as the partial feedback information $\mathbf{Y}$, which reflect the human operators' criteria for anomalies in a specific system. For simplicity, we assume that the user feedback are always accurate, and similar data patterns $(x_i, t_i) \approx (x_j, t_j)$ have the same labels $y_i = y_j$.

III. Algorithm Design

In this section, we propose *ACVAE*, a situation-aware MTS anomaly detection algorithm that incorporates *out-of-band information* through *active learning and contrast VAE-based models* for intelligent operation in large distributed systems. We first introduce the motivation and core idea of *ACVAE* (Section III-A), then propose the two key components of *ACVAE*, contrast detection model (Section III-B) and label query model (Section III-D), finally show how to periodically train *ACVAE* and use the model for online MTS anomaly detection (Section III-E).

A. Motivation and Overview

As discussed in Section II-B, out-of-band information (including background information and feedback information) are important for improving MTS anomaly detection performance. In real-world scenarios, human operators often take both raw MTS data and its corresponding background information into consideration to detect anomalies (*e.g.*, long-periodical anomaly, correlation anomaly). For the unusual MTS data segments, operators utilize feedback information

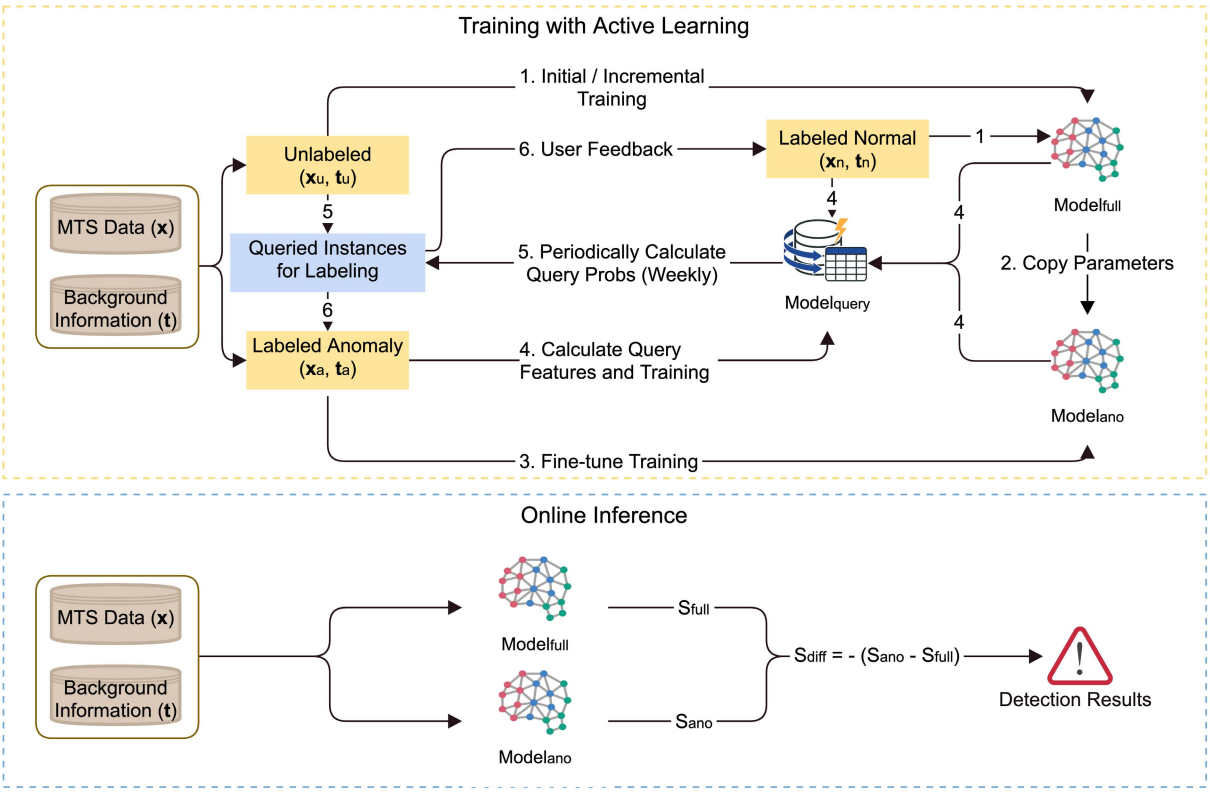


Fig. 2. Overview framework of *ACVAE*. The training procedure loops step 1 to 6 each week to train contrast detection models and the query model with active learning, while the online inference can be triggered once a new data point is collected. The detailed algorithms are shown in Algorithm 1 and Algorithm 2.

(user feedback and domain knowledge) to distinguish if they are only "data anomalies" caused by normal external events (thus, operators are not interested in them and treat them as normal data) or "real anomalies" interested by users (*i.e.*, the anomalies caused by system failures and have negative effects on downstream tasks).

Similarly, the **core idea** of *ACVAE* is to act as the human operators, which *incorporates the out-of-band information into its anomaly detection model, while proactively queries few but important user feedback to reduce human workloads*. In this way, *ACVAE* can gradually learn domain experts' interests for "real anomalies" and give more precise alerts. Fig. 2 shows the overview framework of *ACVAE*.

To tackle challenge 1 in Section I, *ACVAE* trains a pair of contrast models, $\text{model}_{\text{full}}$ and $\text{model}_{\text{ano}}$, to utilize the feedback information about normal and anomalous data. $\text{model}_{\text{full}}$ is trained with labeled normal data (as well as normal data augmentation) and unlabeled data to learn normal MTS patterns, while $\text{model}_{\text{ano}}$ is fine-tuned based on $\text{model}_{\text{full}}$ only with the labeled anomaly data to learn the anomalous MTS patterns for users. In this way, the difference between anomaly scores of $\text{model}_{\text{ano}}$ and $\text{model}_{\text{full}}$ serve as an anomaly indicator for the MTS data. *ACVAE* also uses a conditional input variable and a learnable prior distribution in both contrast detection models to leverage the background information like timestamp and flag series to learn current system status. Moreover, *ACVAE* adopts a non-linear state space model structure in both $\text{model}_{\text{full}}$ and $\text{model}_{\text{ano}}$ to capture the temporal and correlational information inside MTS data. To tackle challenge 2 in Section I, in order to use less feedback to improve the detection performance, we propose a query model that proactively finds the most useful data instances for model training. Specifically, the query model aims at querying more ground-truth anomalies, false positives, and false negatives, whose labels are more valuable for training the detection model. Last but not least, *ACVAE* is trained with an active learning fashion, which makes it continuously learn and adapt to potential new patterns and labels. *ACVAE* also has a low query and incremental training frequency (*e.g.*, once a week) to make it more practical in real-world scenarios.

B. Contrast Detection Models With Active Learning

Recent progresses on MTS anomaly detection are mostly unsupervised methods [2], [3], [12]–[15], [24], which aim at learning the prediction or reconstruction of normal MTS patterns from the historical MTS data. However, without incorporating the out-of-band information, it's often hard for these methods to learn the domain experts' interests for "real anomalies", leading to poor detection performance.

Inspired by recent progress in image out-of-distribution (OoD) detection [25], [26], the improvement of the likelihood for a fine-tuned model on test samples can serve as an effective out-of-distribution indicator. However, for MTS anomaly detection, it's often not possible to fine-tune the model on test data before detection, since we want to get detection

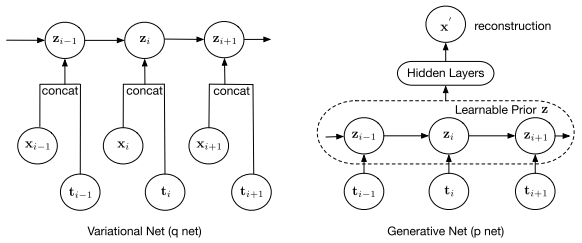


Fig. 3. Graphical model structure of *ACVAE*. $\mathbf{x}$ is the MTS input, $\mathbf{t}$ is the conditional input that represents encoded background information (*e.g.*, timestamp, system flags), $\mathbf{z}$ is the stochastic latent variables.

results in an online fashion (which needs a very short model inference time). Moreover, it's often hard to obtain a pure normal training set as in image OoD problems. This makes it impossible to directly apply their methods on MTS anomaly detection task.

Fortunately, it's possible to make use of the out-of-band information and solve the above problems in MTS anomaly detection. Specifically, we propose **a pair of contrast detection models** based on Variational Auto-Encoder [18] (VAE, a kind of deep generative model) to learn the normal and anomalous patterns for anomaly detection. $\mathrm{model}_{\mathrm{full}}$ is trained with labeled normal data (from feedback information) and other unlabeled training data to learn the normal patterns of MTS, while $\mathrm{model}_{\mathrm{ano}}$ is initialized with the parameters of well-trained $\mathrm{model}_{\mathrm{full}}$ and fine-tuned with the labeled anomalous data from feedback. Note that, all the above models are trained with historical training data in an active learning fashion (as shown in Fig. 2), then the models are used for online anomaly detection. Intuitively, after initialized with the well-trained $\mathrm{model}_{\mathrm{full}}$ and fine-tuned on labeled anomalous data, $\mathrm{model}_{\mathrm{ano}}$ gives much higher likelihood (and also better reconstruction probabilities) on similar anomalous data than $\mathrm{model}_{\mathrm{full}}$, while the likelihood of well-trained normal data changes slightly compared with the anomalies. It's also verified by the empirical results in Fig. 9(a) and Section V-B. This is in accordance with the observations in OoD detection [25], [26], where the fine-tuned model will have a higher likelihood increase on the out-of-distribution samples than in-distribution samples compared with the original model. Therefore, the difference between anomalies scores (*i.e.*, reconstruction probability $\mathbb{E}_{q_\phi(z|x,t)}[\log p_\theta(x|z,t)]$ in our model) of $\mathrm{model}_{\mathrm{ano}}$ and $\mathrm{model}_{\mathrm{full}}$ can serve as an effective anomaly indicator.

The detailed model structure of *ACVAE* is shown in Fig. 3.

To take the *background information* (*e.g.*, time, system flags) into consideration, we make it as the **conditional input** t of *ACVAE* and learn the latent embeddings. Moreover, the **learnable prior distribution** $p_\theta(z)$ is modeled by a non-linear state space model [27], [28] via $p_\theta(z_i|z_{i-1}, t_i)$, which captures both temporal information and background information. For example, as discussed in the Flag case in Section II-B, $t_i = (s^1, s^2, \ldots, s^M), s \in \{0, 1, 2\}$ represents status (no event, code deployment, restart) for different groups of clusters, which is taken into consideration by human operators to get knowledge about current group status and do anomaly detection according to correlations among groups with the same status. Similarly, *ACVAE* trains a learnable prior distribution condition on the input background information to get knowledge about the current system status, which imitates the operators' behavior for anomaly detection.

To model the *temporal and correlational information* of MTS data and learn the normal and abnormal patterns, we also adopt the **state space model** [27], [28] to build a temporal connection between latent stochastic variables z. *i.e.*, $q_\phi(z_i|z_{i-1}, x_i, t_i)$ with trainable parameters ϕ modeled by neural networks, where the low-dimensional latent variables z capture the temporal and correlational dependencies from input MTS x condition on its corresponding background information t. This VAE-based model can be trained in an unsupervised manner, which makes it possible to train $\mathrm{model}_{\mathrm{full}}$ even only with unlabeled data.

To leverage the *feedback information*, we do data augmentation for the labeled normal segments and design an anomaly loss to fine-tune $\mathrm{model}_{\mathrm{ano}}$ on labeled anomalous instances. The whole model is trained in an active learning fashion to continuously learn new patterns and query labels about the most useful instances for the current detection model. Specifically, the query model (described in Section III-D) selects the most useful data segments $x_{i-W+1:i}$ from the unlabeled data and obtains labels from user feedback. Then $\mathrm{model}_{\mathrm{full}}$ can be trained with labeled normal data segments x_{norm} and unlabeled data segments x_U, by optimizing the ELBO [18], [29] in Eq. (1).

$$\begin{aligned}&\mathcal{L}_{full}(\mathbf{x}, \theta, \phi)\\ &\quad = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{t})}\left[\log p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{t})\right] - D_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})||p_\theta(\mathbf{z}|\mathbf{t})\big)\end{aligned} \tag{1}$$

where $D_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})||p_\theta(\mathbf{z}|\mathbf{t})\big) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{t})}(\log q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t}) - \log p_\theta(\mathbf{z}|\mathbf{t}))$ is the Kullback-Leibler divergence [18]. According to Fig. 3, we ignore the future dependency [28] and have $\log q_\phi(\mathbf{z}_{1:W}|\mathbf{x}_{1:W}, \mathbf{t}_{1:W}) = \sum_{i=1}^{W} \log q_\phi(\mathbf{z}_i|\mathbf{z}_{i-1}, \mathbf{x}_i, \mathbf{t}_i)$, $\log p_\theta(\mathbf{z}_{1:W}|\mathbf{t}_{1:W}) = \sum_{i=1}^{W} \log p_\theta(\mathbf{z}_i|\mathbf{z}_{i-1}, t_i)$. The whole objective can be optimized using SGVB estimator and reparameterization trick [18]. To enhance $\mathrm{model}_{\mathrm{full}}$ learning the labeled normal data, we do **normal data augmentation** for the labeled normal data segments x_{norm}, by adding random Gaussian noise (Eq. (2)). Then $\mathrm{model}_{\mathrm{full}}$ is incrementally trained with (x_U, x_{norm}, x_{aug}) after obtained each week's query labels. Since queried x_{norm} are mostly false positives for current $\mathrm{model}_{\mathrm{full}}$, training on augmented x_{norm} can help the model pay more attention on learning such normal patterns to reduce the number of false positives.

$$x_{aug} = x_{norm} + \tilde{x}, \quad \tilde{x} \sim \mathcal{N}(0, \epsilon) \tag{2}$$

After $\mathrm{model}_{\mathrm{full}}$ is well-trained to learn the normal patterns, $\mathrm{model}_{\mathrm{ano}}$ is initialized with the parameters of $\mathrm{model}_{\mathrm{full}}$ to obtain knowledge about current normal patterns. As a likelihood model, **fine-tuning** $\mathrm{model}_{\mathrm{ano}}$ **on labeled anomalous data** x_{ano} will improve its likelihood on similar anomalous data, while its likelihood on other normal data remains similar with $\mathrm{model}_{\mathrm{full}}$. To improve model training with limited x_{ano}, we use each input window that contains at least ONE *queried anomalous point* as the training data, but only pay attention

to learning the anomalous parts by optimizing $\mathcal{L}_{ano}(\mathbf{x}, \theta, \phi)$ in Eq. (3).

$$\begin{aligned}&\mathcal{L}_{ano}(\mathbf{x}, \theta, \phi)\\&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{t})}\left[\mathbf{y} * \log p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{t})\right] - D_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{t})||p_\theta(\mathbf{z}|\mathbf{t})\big)\end{aligned} \tag{3}$$

where $y_i = 1$ for queried anomalous points, $y_i = 0$ for others. Through the fine-tune training, $\mathrm{model}_{\mathrm{ano}}$ tries to distinguish different anomalous patterns from the normal patterns and assign the anomalies higher likelihoods (also higher reconstruction probabilities $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{t})}\left[\log p_\theta(\mathbf{x}_{ano}|\mathbf{z}, \mathbf{t})\right]$). Therefore, although $\mathrm{model}_{\mathrm{ano}}$ may not be able to faithfully reconstruct each anomalous pattern with limited training data, the difference between anomaly scores given by $\mathrm{model}_{\mathrm{ano}}$ and $\mathrm{model}_{\mathrm{full}}$ can serve as a reliable anomaly detection indicator.

C. Anomaly Score

Previous works [2], [3] have shown that the reconstruction probability $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right]$ is an effective anomaly score for unsupervised MTS anomaly detection methods. Therefore, in *ACVAE*, we use the reconstruction probability S_{full} and S_{ano} as the score of our contrast detection models (Eq. (4) and Eq. (5)). As discussed in Section III-B, the difference between S_{ano} and S_{full} can serve as an effective anomaly indicator for *ACVAE*. We denote the anomaly score as S_{diff}, as shown in Eq. (6). In order to detect anomalies in time, we choose the sliding window with time index $(i - W + 1, \ldots, i - 1, i)$ for detecting anomaly at index i, and use the score for the **last data** $\mathbf{x}_i$ in the window as the detection result. The instances with **lower** S_{diff} are more likely to be anomalies.

$$S_{full} = \mathbb{E}_{q_{\phi_{full}}(\mathbf{z}|\mathbf{x},\mathbf{t})}\left[\log p_{\theta_{full}}(\mathbf{x}|\mathbf{z}, \mathbf{t})\right] \tag{4}$$

$$S_{ano} = \mathbb{E}_{q_{\phi_{ano}}(\mathbf{z}|\mathbf{x},\mathbf{t})}\left[\log p_{\theta_{ano}}(\mathbf{x}|\mathbf{z}, \mathbf{t})\right] \tag{5}$$

$$S_{diff} = -(S_{ano} - S_{full}) \tag{6}$$

Single-model anomaly detection methods [2], [3] only learn the normal patterns and use S_{full} as the anomaly score. However, with the help of contrast detection models, *ACVAE* learns both normal and abnormal patterns from out-of-band information and uses S_{diff} as the anomaly score, which reduces the number of FPs and FNs for MTS anomaly detection. We conduct ablation studies in Section IV-F.1 to quantitatively show the effectiveness of using S_{diff} and contrast detection models, and use case studies in search backend systems (Section V-B) to show how *ACVAE* precisely detects anomalies in different situations.

D. Label Query Model for Active Learning

In order to train *ACVAE* with as little user feedback as possible to reduce human workloads, we propose a label query model to proactively find the most useful data to obtain user feedback. The query model is trained based on current detection models' outputs and historical labels. The design of the query model is composed of three parts: feature selection, loss design, and model initialization.

In active learning, the query model Q uses the features of the current detection model $\mathcal{D}$ as its input. Then Q generates queries x_Q to find the most useful unlabeled data instances and interact with human experts. Finally, the ground-truth labels y_Q are provided by human experts after investigation. (x_Q, y_Q) and other unlabeled data are then used to optimize the detection model $\mathcal{D}$ for better performance. Specifically, for MTS anomaly detection problem, we mainly focus on the rare anomalous instances, and the FP and FN instances where $\mathcal{D}$ has made a mistake. The feedback labels for these instances are more useful for improving our detection model, thus are the aim of the query model.

1) Feature Selection: Following the above criteria, we mainly extract features that can represent potential anomalies and model's uncertain instances from the current detection model $\mathcal{D}$ as the input of query model Q. Specifically, to capture potential anomalies, we directly use the anomaly score S_{full} for $\mathrm{model}_{\mathrm{full}}$ (Eq. (4)) and S_{diff} for contrast model (Eq. (6)) as the feature, since they directly represent the detection result of current detection model $\mathcal{D}$. To find the uncertain instances of $\mathcal{D}$, we take the standard deviations of posterior distribution at each point x_i, *i.e.*, σ_{full_i} for $\mathrm{model}_{\mathrm{full}}$ (Eq. (7)) and σ_{anoi} for $\mathrm{model}_{\mathrm{ano}}$ (Eq. (8)) as features. In MTS anomaly detection, the anomalies often last for some time and form an anomaly segment. Intuitively, the most uncertain instances for detection model $\mathcal{D}$ often lie in the sliding window $[x_{norm}, x_{ano}]$, where the MTS change from normal patterns to anomalous ones, or vice versa. When unlabeled instance $[x_{norm}, x'_{norm}]$ (normal) and $[x_{norm}, x_{ano}]$ (anomaly) are trained by $\mathcal{D}$, the model will learn large standard deviations for $p_\theta(x|z)$ on x'_{norm} and x_{ano}, which indicates $\mathcal{D}$ is not sure how to reconstruct these instances based on previous observation x_{norm}. Oppositely, the standard deviation will be small for the well-learned instances, whose detection results are convinced by $\mathcal{D}$. Therefore, σ_{full_i} and σ_{anoi} are used to capture the uncertainty of the detection model. The input feature vector at each x_i for query model is shown as Eq. (9). We also show experimental results in Section IV-F.2 to demonstrate the effectiveness of our selected features on finding the anomalous and uncertain instances.

$$p_{\theta_{full}}(\mathbf{x}_i|\mathbf{z}, \mathbf{t}) \sim \mathcal{N}(\mu_{full_i}, \sigma_{full_i}) \tag{7}$$

$$p_{\theta_{ano}}(\mathbf{x}_i|\mathbf{z}, \mathbf{t}) \sim \mathcal{N}(\mu_{anoi}, \sigma_{anoi}) \tag{8}$$

$$V_i = (S_{full_i}, S_{diff_i}, \sigma_{full_i}, \sigma_{anoi}) \tag{9}$$

2) Loss Design: The query model Q aims to query the ground-truth anomalies and FP, FN points for current detection model $\mathcal{D}$. Since we do not know the labels of unlabeled x_U, the query model is *only trained with past queries* x_{norm} *and* x_{ano} *with known labels*. Let q be the set of past queried point indexes. f is composed of several fully-connected layers [30] with tanh activation [31], and a sigmoid activation [30] is applied on the last layer to obtain a query probability:

$$\begin{aligned}\mathrm{prob} &= [f(V_i), i \in q]\\ y_g &= [y_i, i \in q]\\ \mathrm{fpfn} &= [\mathrm{pred}_i \oplus y_i, i \in q]\end{aligned} \tag{10}$$

where y is the ground-truth label of past queried points, pred is the current detection result of *ACVAE*. We use binary cross-entropy (BCE) loss [30], [32] to train the query model using past queries with feedback labels. The total loss $\mathcal{L}_{query}$ of query model is shown in Eq. (11), where β is a hyperparameter to balance the weight of learning to query ground-truth anomalies and query current FP and FN points. In this paper we empirically set $\beta = 0.25$.

$$\begin{aligned} \mathcal{L}_g &= BCE(y_g, \text{prob}) \\ \mathcal{L}_f &= BCE(\text{fpfn}, \text{prob}) \\ \mathcal{L}_{query} &= \mathcal{L}_g + \beta * \mathcal{L}_f \end{aligned} \tag{11}$$

3) Model Initialization: To generate more accurate queries at the beginning of training, we use a few labeled anomaly data to initialize the query model. Specifically, in the industrial monitoring system, system operators write incident reports for past severe anomalies. Therefore, in this paper, we use several anomalous cases (1 to 5 cases for different systems) collected from past incident reports as an initialization. In case there is no labeled anomalous case for a system, the query model will first greedily query data instances with lower anomaly scores (which are more likely to be anomalies) as an initialization, until it successfully queries a ground-truth anomaly segment. Moreover, an epsilon greedy strategy [33] can be applied to such a scenario to improve exploration.

E. Model Training and Inference

As shown in Fig. 2, *ACVAE* is trained with active learning. **a) Model Initialization.** Both the detection model $\text{model}_{\text{full}}$, $\text{model}_{\text{ano}}$ and query model Q are initialized with several anomalous cases from past incident reports and initial unlabeled data (including raw MTS and their corresponding background information). Then the query model is applied on unlabeled data instances and selects the most useful data to obtain user feedback. **b) Active Training.** 1. $\text{model}_{\text{full}}$ is trained with unlabeled data and labeled normal data to learn the normal patterns. 2. Synchronize network parameters from $\text{model}_{\text{full}}$ to $\text{model}_{\text{ano}}$ for initialization, and make $\text{model}_{\text{ano}}$ learn *current* normal patterns. 3. Fine-tune training $\text{model}_{\text{ano}}$ with labeled anomalous data to learn anomalous patterns. 4. Calculate query features for past queried instances with the updated detection models, then train the query model with these instances. 5. Apply the query model on unlabeled data to obtain query probabilities and select new data instances to be queried. 6. Ask for user feedback to obtain labels for queried instances. The model training (steps 1 to 6 above) is done periodically. In this paper, the training is triggered once a full week's data (in the training set) are collected. And the query model queries 20 data windows each week, until the feedback reaches the total query portion (3%). **c) Online Inference.** For online inference, once a new data point is collected, *ACVAE* looks ahead 30 points (*i.e.*, a 30-minute window in our data) to form a sliding window and detect if the latest point is an anomaly or not. Points with lower scores (Eq. (6)) are more likely to be anomalies. Algorithm 1 and Algorithm 2 show the training and inference pipeline of *ACVAE*.

Algorithm 1 *ACVAE* Training

Input: Unlabeled MTS $\mathbf{x}$ and its corresponding background information $\mathbf{t}$, initial anomalous cases MTS $\mathbf{x}_{init}$ and corresponding background information $\mathbf{t}_{init}$, label $\mathbf{y}_{init}$

Output: The trained detection models $\text{model}_{\text{full}}$ (with parameters θ_f and ϕ_f) and $\text{model}_{\text{ano}}$ (with parameters θ_a and ϕ_a)

Unlabled Set $\mathbf{x}_U = \mathbf{x}$, Labeled normal set $\mathbf{x}_{norm} = \{\}$, Labeled anomalous set $\mathbf{x}_{ano} = \mathbf{x}_{init}$
Train $\text{model}_{\text{full}}$ (θ_f, ϕ_f) using Eq. (1) with $(\mathbf{x}_U, \mathbf{t}_U)$.
$(\theta_a, \phi_a) = (\theta_f, \phi_f)$ (copy parameters for initialization).
Fine-tune $\text{model}_{\text{ano}}$ (θ_a, ϕ_a) using Eq. (3) with initial anomalous cases $(\mathbf{x}_{init}, \mathbf{t}_{init}, \mathbf{y}_{init})$.
for week $w = 1, 2, \ldots$ until training data is used up **do**
  Extract features V_i (Eq. (9)) for labeled instances $\mathbf{x}_{norm}$ and $\mathbf{x}_{ano}$, train query model Q using Eq. (11).
  Obtain new data for week w: incremental MTS data $\mathbf{x}_w$ and corresponding background information $\mathbf{t}_w$. $\mathbf{x}_U = \mathbf{x}_U \cup \mathbf{x}_w$, $\mathbf{t}_U = \mathbf{t}_U \cup \mathbf{t}_w$.
  Extract features V_i (Eq. (9)) for $\mathbf{x}_U$.
  Calculate query probabilities $prob$ for $\mathbf{x}_U$ using Q (Eq. (10)).
  Query k unlabeled data segments with highest $prob$ and obtain user feedback $(\mathbf{x}'_{norm}, \mathbf{y}'_{norm})$, $(\mathbf{x}'_{ano}, \mathbf{y}'_{ano})$.
  $\mathbf{x}_{norm} = \mathbf{x}_{norm} \cup \mathbf{x}'_{norm}$, $\mathbf{x}_{ano} = \mathbf{x}_{ano} \cup \mathbf{x}'_{ano}$, $\mathbf{x}_U = \mathbf{x}_U \setminus (\mathbf{x}'_{norm} \cup \mathbf{x}'_{ano})$.
  Incremental train $\text{model}_{\text{full}}$ (θ_f, ϕ_f) using Eq. (1) with $(\mathbf{x}_U, \mathbf{x}_{norm}, \mathbf{t}_U, \mathbf{t}_{norm})$.
  $(\theta_a, \phi_a) = (\theta_f, \phi_f)$ (synchronize parameters).
  Fine-tune $\text{model}_{\text{ano}}$ (θ_a, ϕ_a) using Eq. (3) with labeled anomalies $(\mathbf{x}_{ano}, \mathbf{t}_{ano}, \mathbf{y}_{ano})$.
end for
return $\text{model}_{\text{full}}$ (θ_f, ϕ_f), $\text{model}_{\text{ano}}$ (θ_a, ϕ_a)

IV. Experiments and Analysis

In this section, we first introduce the datasets, comparison methods and metrics for evaluating *ACVAE*, then design experiments to answer the following research questions:

RQ1: How does *ACVAE* perform on MTS anomaly detection in different situations?

RQ2: How much performance gain on MTS anomaly detection can *ACVAE* get from incorporating out-of-band information?

RQ3: How effective is the query model of *ACVAE* on finding the most useful instances for MTS anomaly detection?

RQ4: How effective is each design of *ACVAE*?

A. Dataset

To demonstrate the effectiveness of *ACVAE* in a real production environment, we collected three types of MTS data from search backend systems at eBay: cluster-level metrics, group-level metrics, and data center level metrics. The metrics are collected from different machines in the distributed systems and aggregated to different levels, which reflects the resource usage, network conditions, *etc*. Operators monitor the systems'

Algorithm 2 *ACVAE* Online Inference

Input: Unlabeled MTS $\mathbf{x}$ and its corresponding background information $\mathbf{t}$, trained detection models $\mathrm{model}_{\mathrm{full}}$ and $\mathrm{model}_{\mathrm{ano}}$.

Output: Anomaly score list S_{diff}.

1: **for** new coming point $(\mathbf{x}_i, \mathbf{t}_i) \in (\mathbf{x}, \mathbf{t})$ **do**
2: Look ahead 30 points to form the detection window $(\mathbf{x}_{i-29:i}, \mathbf{t}_{i-29:i})$ of $\mathbf{x}_i$.
3: Calculate S_{full_i} for $\mathbf{x}_i$ using $\mathrm{model}_{\mathrm{full}}$ and Eq. (4).
4: Calculate S_{anoi} for $\mathbf{x}_i$ using $\mathrm{model}_{\mathrm{ano}}$ and Eq. (5).
5: $S_{diff_i} = -(S_{anoi} - S_{full_i})$.
6: Append S_{diff_i} to S_{diff}. Data instances with smaller S_{diff} are more anomalous.
7: **end for**
8: **return** S_{diff}

health status according to these metrics and other out-of-band information. We briefly describe each dataset as follows:

1. Cluster-Level Metrics (CLM): At eBay, large amounts of applications are running on the application clusters. For different applications, the metrics might include CPU usage, QPS, memory usage, latency, error counts, etc, which are used to characterize the state of the cluster. We collected these key cluster metrics to monitor the health of the application clusters. In CLM, regular jobs are performed with different periodicities, and the period-to-period basis is concerned by human operators for anomaly detection. Therefore, the timestamp is collected as background information for CLM data. In this paper, the timestamp is encoded to a 91-dimension one-hot vector (which represents day-of-week, hour-of-day, minute-of-hour) and used as input background information $\mathbf{T}$.

2. Group-Level Metrics (GLM): Large applications often have a large number of nodes. In order to ensure service continuity, these nodes are grouped together, and the deployment is done group by group. GLM contains system metrics collected from each group of the application. We collected group metrics for applications that require frequent changes to detect issues that occur during application deployment. In GLM, human operators are concerned about the group status and MTS metrics for anomaly detection. Therefore, we collect the flag series, which is recorded by the deployment system to show the status of each group (no event, code deployment, and restart), as background information for GLM data. The flag series is already a real-valued vector, thus is directly used as input background information $\mathbf{T}$.

3. Data Center Level Metrics (DCLM): At eBay, each application is deployed to multiple data centers at the same time for high availability. Under the distribution of the load balancing system, the same metrics of the same application in different data centers have a certain correlation characteristic, and DCLM refers to the same metric of the same application on different data centers. We collected CPU usage and QPS metrics of large applications in different data centers for monitoring the health of data centers as well as load balancers. In DCLM, the metrics often do NOT have periodicity, and the human operators concern more about the "normal patterns" of correlations among metrics in different data centers. There is no background information for DCLM since the operators directly detect anomalies based on the raw MTS data.

TABLE I
DATASET STATISTICS

Dataset	# metrics	Total points	Anomaly (%)
CLM	44	259201	0.3144
GLM	7	138241	0.8450
DCLM-1	6	137990	8.1129
DCLM-2	6	137005	7.4275
SKAB	8	46860	28.2565

Moreover, domain knowledge and the effect on downstream business also affect the operators' anomaly detection criteria in different systems. Therefore, it's important to incorporate feedback information for anomaly detection in the above systems. The dataset statistics are shown in Table I. The CLM dataset was collected for 6 months, while other datasets were collected for 3 months. The data collection interval for each dataset is 1 minute per point. For each dataset, we use the first 50% data (in time order) as the training set, while the last 50% data is the testing set. During data collection, we buried points in the system and exposed APIs to collect key performance metrics (*e.g.*, CPU usage, network delay). Beats [34] system, a data collector, called the API and collected data points with a fixed time interval, then stored the data points in Prometheus [35] to form time series. Average or sum aggregation was performed on different metrics using the functions provided by Prometheus [35]. CLM dataset also contains the Mediff [4] feature extracted from raw time series data. The background information is extracted from structured system logs using simple scripts. We collect the ground-truth labels according to the actual issue tickets in production, as well as the discussion between system operators and downstream business users. We listed some typical normal and abnormal events in different systems as examples in Table II.

Moreover, we also use a public MTS anomaly detection dataset SKAB [36] to evaluate *ACVAE*'s detection performance in other domains. SKAB [36] is a sensor dataset about water circulation. It contains one training file with pure normal data, and 35 files with different kinds of labeled anomalies and other normal data. In order to provide anomaly labels for semi-supervised and supervised algorithms, we use 7 files with anomalies for training, and keep the other 28 files as the testing set. SKAB dataset does not have background information.

B. Comparison Methods

To demonstrate the effectiveness of *ACVAE*, we compare its anomaly detection performance with twelve comparison methods, including AutoEncoder [37], Isolation Forest [38], USAD [15], InterFusion [3], UAE-GD [16], GDN [17], LSTM-NDT [12], Ymir [39], TapNet [40], meta-AAD [41], Devnet [42] and DeepSAD [43].

1) Unsupervised Methods: **AutoEncoder** [37] and **Isolation Forest** [38] are two popular anomaly detection methods, which mainly use the effect of dimension reduction and isolation to detect anomalies, respectively. **LSTM-NDT** [12] makes

TABLE II
EXAMPLES OF TYPICAL NORMAL AND ABNORMAL EVENTS IN DIFFERENT PRODUCTION SYSTEMS

Dataset	Normal Events	Abnormal Events
CLM	1. The service code changes from time to time. 2. The service performs regular tasks. 3. Regular requests access the service. 4. Code changes during regular task execution. 5. Manually service restart.	1. Regular jobs stopped several times from time to time. 2. Deploying versions of code with lots of errors. 3. Network congestion in some monitored clusters. 4. Network errors increase caused by spurt of requests from upstream.
GLM		1. Deployments are done without writing event signals. 2. Some groups deploy a different code version. 3. Some groups fail to deploy new code then fall back.
DCLM		1. Requests are not distributed to some data centers. 2. Shut down an upstream service in some data centers. 3. Deployment is not applied for clusters in a certain data center.

MTS predictions with LSTM and nonparametric dynamic thresholding for anomaly detection. **GDN** [17] takes advantage of graph neural networks to model the correlations inside MTS data for anomaly detection. **USAD** [15], **InterFusion** [3], and **UAE-GD** [16] are three state-of-the-art unsupervised MTS anomaly detection methods based on deep learning, which have obtained the best performance on several unsupervised MTS anomaly detection datasets [3], [15], [16]. Taking advantage of adversarial training and two-view embedding help USAD and InterFusion learn the normal pattern of MTS from historical unlabeled MTS data. UAE-GD achieves high detection performance using its dynamic Gaussian scoring function.

2) Semi-Supervised Methods: Meta-AAD [41], Devnet [42] and DeepSAD [43] are three state-of-the-art semi-supervised general anomaly detection methods. Here we apply these methods to multivariate time series data. Specifically, **meta-AAD** [41] is an active anomaly detection algorithm. It proposed to use deep reinforcement learning [44] to train a meta-policy and select the most useful instances to query. Meta-AAD also claimed that the learned meta-policy could be transferred among different datasets. However, in this paper, we train a meta-AAD model for each dataset to achieve the best performance. **Devnet** [42] is a semi-supervised anomaly detection method that directly optimizes anomaly scores with labeled anomaly data. **DeepSAD** [43] is a semi-supervised anomaly detection method that leverages an information-theoretic idea to split normal and anomalous data in the latent space from labeled normal and anomalous data. Devnet and DeepSAD do not have a query model to find the useful points, so we randomly select normal and anomalous data according to the anomaly ratio of each dataset as their input labels. Moreover, the same initial anomalous cases collected from past incident reports (Section III-D) are used for all the semi-supervised and supervised methods.

3) Supervised Methods: **TapNet** [40] is a state-of-the-art supervised algorithm for MTS classification. Here we treat the anomaly detection as a binary classification problem and apply TapNet for comparison. The classification probabilities are used as the anomaly score for evaluation. **Ymir** [39] is an ensemble model for MTS anomaly detection, which is developed by domain experts and deployed in the production system of eBay. Ymir integrates several unsupervised models, including Mediff detector [4], Chebyshev theorem [45], Extreme Value Theory [46], moving average, Variational AutoEncoder [18] and isolation forest [38]. Ymir normalizes the decision boundaries of the above 6 unsupervised models into the corresponding 6 groups of anomaly scores, which contains rich potential information to detect anomalies, like periodic bias, local bias, distribution bias, etc. In an unsupervised setting, Ymir uses the weighted sum of the aforementioned 6 groups of anomaly scores as the final detection score. In a supervised setting, the 6 groups of anomaly scores, the raw MTS data, the background information and corresponding labels are used as input for training a transformer-based classifier. Besides, a semi-supervised Ymir uses the partial labels to revise the unsupervised feature extraction results and train the classifier for anomaly detection.

C. Evaluation Metrics

For MTS data in industrial monitoring systems, anomalies often last for a while and form a contiguous anomaly range. The anomaly detection algorithm also raises one alert for each detected anomaly range (*i.e.*, range-based anomalies [16], [47]). To evaluate the anomaly detection performance on such data, we use the well-known range-based precision (recall, F1-score) [47] that is specifically designed for evaluating the range-based anomaly detection on time series data.

Let $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_{N_\mathcal{P}}\}$ denote the set of predicted anomaly ranges given by detection model $\mathcal{D}$, $\mathcal{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_{N_\mathcal{R}}\}$ denote the set of ground-truth anomaly ranges in test set. Then the range-based recall $Recall_T(\mathcal{R}, \mathcal{P})$ can be calculated as Eq. (12), which considers the existence, size, position and cardinality for anomaly detection on time series data [47]. $N_\mathcal{R}$ is the total number of ground-truth anomaly ranges. $N_\mathcal{P}$ is the number of detected anomaly ranges given by detection model $\mathcal{D}$.

$$\begin{aligned} Recall_T(\mathcal{R}, \mathcal{P}) &= \frac{\sum_{i=1}^{N_\mathcal{R}} Recall_T(\mathcal{R}_i, \mathcal{P})}{N_\mathcal{R}} \\ Recall_T(\mathcal{R}_i, \mathcal{P}) &= \alpha \times ExistenceReward(\mathcal{R}_i, \mathcal{P}) \\ &\quad + (1-\alpha) \times CardFactor(\mathcal{R}_i, \mathcal{P}) \\ &\quad \times \sum_{j=1}^{N_\mathcal{P}} \omega(\mathcal{R}_i, \mathcal{R}_i \cap \mathcal{P}_j, \delta_r) \end{aligned} \tag{12}$$

Specifically, for anomaly detection in industrial systems, we want the anomalies to be detected as early as possible, since late detection often has little help for preventing financial loss. Therefore, *for a given anomaly range $\mathcal{R}_i$, we let*

$ExistenceReward = 1$ *only if at least one point in the first 10 points of* $\mathcal{R}_i$ *is detected by* $\mathcal{P}$. The existence reward is similar to the adjust score in [7] and Fc_1 score in [16], but takes the detection delay into consideration to meet the requirements in practice. We empirically set $\alpha = 0.8$ to pay more attention to detecting each anomaly range. If a ground-truth anomaly range $\mathcal{R}_i$ is overlapped with k $\mathcal{P}_j \in \mathcal{P}$, we set $CardFactor(\mathcal{R}_i, \mathcal{P}) = \frac{1}{k^{0.2}}$ to encourage detecting the anomaly with less prediction ranges. ω term is used to calculate the overlap size and position reward. Specifically, for a given anomaly range $\mathcal{R}_i$, the more points in $\mathcal{R}_i$ are detected, the better the detection model is. Moreover, $\delta_r(idx)$ can be regarded as the weight for detecting point at position idx in $\mathcal{R}_i$. We use a decay weight for calculating recall since the earlier points in an anomaly range are more important. $l(\mathcal{R}_i)$ is the length of anomaly range $\mathcal{R}_i$.

$$\omega(\mathcal{R}_i, \mathcal{R}_i \cap \mathcal{P}_j, \delta_r) = \frac{\sum_{idx \in \mathcal{R}_i \cap \mathcal{P}_j} \delta_r(idx, l)}{\sum_{idx=1}^{l} \delta_r(idx, l)}$$
$$\delta_r(idx, l) = (l^2 - (idx - 1)^2)^{0.5}$$

Similarly, the range-based precision $Precision_T(\mathcal{R}, \mathcal{P})$ can be calculated using Eq. (13). $W_{\mathcal{P}_i} = \log_2[l(\mathcal{P}_i) + 1]$ is the weight to balance different prediction lengths. Since the point position in detected $\mathcal{P}_i$ has no effect on the precision, we set $\delta_p = 1$ for all points.

$$Precision_T(\mathcal{R}, \mathcal{P}) = \frac{\sum_{i=1}^{N_\mathcal{P}} Precision_T(\mathcal{R}, \mathcal{P}_i) W_{\mathcal{P}_i}}{\sum_{i=1}^{N_\mathcal{P}} W_{\mathcal{P}_i}}$$
$$Precision_T(\mathcal{R}, \mathcal{P}_i) = CardFactor(\mathcal{R}, \mathcal{P}_i) \times \sum_{j=1}^{N_\mathcal{R}} \omega(\mathcal{P}_i, \mathcal{P}_i \cap \mathcal{R}_j, \delta_p)$$

Similar to standard precision and recall, $Precision_T$ and $Recall_T$ range from 0 to 1. The range-based F1-score can be calculated using Eq. (13). The range F1 score balanced the precision and recall, which is used as our main evaluation metric. The higher range F1 means the detection model is better at anomaly detection.

$$F1_T = 2 \times \frac{Precision_T \times Recall_T}{Precision_T + Recall_T} \tag{13}$$

Moreover, following previous works [2], [3], [13], [15] in MTS anomaly detection, we enumerate and find the optimal threshold to convert anomaly scores given by detection model $\mathcal{D}$ to detection set $\mathcal{P}$, and then calculate the best range-based F1-score. This makes our evaluation compare the performance of different anomaly detection algorithms regardless of different threshold selection criteria. More details of range-based evaluation on time series data can be found in [47]. We also use the AUPRC, *i.e.*, the area under range-based Precision-Recall curve as an evaluation metric. AUPRC reflects the algorithms' anomaly detection performance without selecting the best threshold.

D. Experiment Setup

The preprocessing and key configurations of *ACVAE* are set as follows. We set the window length $W = 30$. For the neural networks in variational net and generative net (except output layer), we use dense layers with 100 hidden units and tanh activation function. We apply L2 regularization with a coefficient of 10^{-4} to these layers. We set $\epsilon = 0.01$ for normal data augmentation. The log standard deviations of latent distributions and posteriors are clipped within $[-5, 2]$ to avoid numerical problems. We set the latent dimension of $\mathbf{z}$ as 5 for CLM and 4 for other datasets. The algorithm is trained with Adam optimizer [48] with initial learning rate 10^{-3} for $\mathrm{model}_{\mathrm{full}}$ and 10^{-4} for fine-tuning $\mathrm{model}_{\mathrm{ano}}$. We normalize each MTS metric in training set to $[-1, 1]$ with MinMax Scaler. The scaler is further used to preprocess the test data. The last 30% data in the training set is held for validation. For each iteration in Algorithm 1, $\mathrm{model}_{\mathrm{full}}$ is trained for 20 epochs with early stopping, $\mathrm{model}_{\mathrm{ano}}$ is fine-tuned for 50 epochs. The batch size is set to 128 for training. The number of $\mathbf{z}$ samples is set to 100 for Monte Carlo integration to evaluate the reconstruction probability during testing. 5, 3, and 1 anomalous case from past incident reports on CLM, GLM, and DCLM datasets are used for initialization in all the semi-supervised and supervised algorithms. In SKAB dataset, the first anomaly window (30 points) in the training set is used for initializing all the semi-supervised and supervised algorithms.

E. Results and Analysis

1) RQ1. Overall Performance: We compare *ACVAE*'s performance with twelve baselines described in Section IV-B. For semi-supervised methods, the user feedback (*i.e.*, labels) are fed into the algorithms according to their own query model (as described in Section IV-B), and the total label portion is about 3% of training data. For supervised models, all the labels in the training set are used. The best range F1 scores are shown in Table III. The range-based AUPRCs are shown in Table IV.

Overall, *ACVAE* performs the best on all compared datasets among unsupervised and semi-supervised methods. Moreover, *ACVAE* with only 3% labels achieves similar or even better results than the supervised methods, which demonstrates the superiority of *ACVAE*'s designs on utilizing out-of-band information for MTS anomaly detection.

a) Unsupervised methods: As shown in Table III, the unsupervised methods, including the recent state-of-the-art MTS anomaly detection methods USAD [15], InterFusion [3] and UAE-GD [16], achieve poor anomaly detection performance. These methods by design do NOT leverage the out-of-band information and only learn the "normal patterns" from raw MTS data. One thing to mention is that, as discussed in Section IV-A, the main challenge in the DCLM dataset is to model the correlations among metrics, which happens to meet the design of USAD, InterFusion and GDN. Therefore, these three methods achieve acceptable performance on DCLM without labels. In general, the unsupervised methods may perform well on data that fit their designs and assumptions, but cannot leverage out-of-band information and learn the domain experts' criteria about anomalies in different application scenarios. Overall, ***ACVAE*** **outperforms the best-performing unsupervised methods by 22.7% to**

TABLE III

BEST RANGE F1 SCORE (F1), RANGE PRECISION (P), AND RANGE RECALL (R) FOR *ACVAE* AND BASELINES. THE RESULTS ON THE DCLM DATASET ARE THE AVERAGE PERFORMANCE METRICS FOR TWO DCLM DATASETS. "BG" MEANS WHETHER THE METHOD USES BACKGROUND INFORMATION OR NOT. "FB" MEANS WHETHER THE METHOD USES FEEDBACK INFORMATION (LABELS) OR NOT. "QM" MEANS WHETHER THE METHOD HAS DESIGNED A QUERY MODEL TO ASK FOR LABELS. SPECIFICALLY, YMIR (100%) AND TAPNET (100%) ARE SUPERVISED METHODS, THUS THEY DO NOT NEED A QUERY MODEL (N/A). THE TOP-3 PERFORMANCE ON EACH DATASET ARE MARKED IN BOLD. WE ALSO ENHANCE THE SEMI-SUPERVISED BASELINE METHODS WITH DESIGNS TO INCORPORATE BACKGROUND INFORMATION AND QUERY MODELS. THE RESULTS ARE SHOWN IN TABLE V AND FIG. 5

BG	FB	QM	Methods (label portion)	CLM			GLM			DCLM		
				F1	P	R	F1	P	R	avg.F1	avg.P	avg.R
×	×	×	AutoEncoder [37]	0.1224	0.0727	0.3846	0.0709	0.0377	0.5821	0.521	0.5545	0.5921
×	×	×	Isolation Forest [38]	0.0813	0.0433	0.6565	0.0462	0.0236	0.9615	0.2867	0.2054	0.5231
×	×	×	USAD [15]	0.1485	0.0839	0.6467	0.1264	0.1098	0.1488	0.7099	0.8636	0.6031
×	×	×	InterFusion [3]	0.3197	0.2826	0.3680	0.1558	0.1009	0.3416	0.7651	0.8275	0.7115
✓	×	×	Ymir (0%) [39]	0.2692	0.1780	0.5519	0.0665	0.0348	0.7692	0.4203	0.4386	0.4447
×	×	×	UAE-GD [16]	0.2473	0.1475	0.7648	0.0650	0.0336	0.9862	0.5272	0.6308	0.5390
×	×	×	GDN [17]	0.1843	0.1066	0.6798	0.0736	0.0386	0.7831	**0.7806**	0.9671	0.6545
✓	×	×	LSTM-NDT [12]	0.0322	0.0167	0.3939	0.0817	0.0452	0.4187	0.2038	0.1139	0.9966
✓	✓	N/A	Ymir (100%) [39]	**0.9033**	1.0000	0.8237	**0.6552**	1.0000	0.4873	**0.9721**	0.9934	0.9518
✓	✓	N/A	TapNet [40]	0.5209	0.7898	0.3885	0.1644	0.1333	0.2144	0.5434	0.7611	0.4260
×	✓	×	Devnet (3%) [42]	0.0327	0.0187	0.1282	0.1596	0.1184	0.2448	0.6392	0.8389	0.6253
×	✓	×	DeepSAD (3%) [43]	0.0483	0.3227	0.0261	0.1177	0.0749	0.2748	0.6151	0.5479	0.7553
×	✓	✓	meta-AAD (3%) [41]	**0.5375**	0.4239	0.7343	**0.1748**	0.1493	0.2110	0.6869	0.8631	0.5976
✓	✓	×	Ymir (3%) [39]	0.3402	0.2852	0.4214	0.1355	0.0798	0.4463	0.6222	0.6362	0.6190
✓	✓	✓	***ACVAE* (ours)** (3%)	**0.9638**	0.9794	0.9488	**0.6808**	0.6920	0.6701	**0.9580**	0.9645	0.9518

337%, which demonstrates the importance of incorporating out-of-band information into situation-aware MTS anomaly detection.

b) Semi-supervised methods: **Devnet** [42] and **DeepSAD** [43] do NOT have designs to use the background information, which downgrades their performance on CLM and GLM datasets, where the background information is also important for human operators to detect anomalies. Moreover, these two methods do NOT have a query model to find the most useful instances, but randomly select labeled normal and anomalous data according to the anomaly ratio of each dataset as the feedback information. However, in the unbalanced anomaly detection data, it's hard for the random query to find the rare anomalous instances with a small label portion (3%), which makes their performance poor. The semi-supervised version of **Ymir** [39] also does NOT have a query model. Moreover, it utilizes the feedback information through a neural network classifier, whose performance significantly drops without sufficient labels. **Meta-AAD** [41] is also an active learning based anomaly detection method, which trains meta-policy using reinforcement learning as their query model. Note that, besides 3% of training labels, we use 3% *test labels* (which should be invalid in training) queried by meta-AAD to fine-tune its features for meta-policy (as required by meta-AAD algorithm [41]). However, it's also hard to perfectly train its reinforcement learning method with such few feedback labels, which makes the performance of meta-AAD falls behind *ACVAE* on the MTS datasets. Moreover, meta-AAD also lacks designs for utilizing the background information. For a more fair comparison, we also **enhance** the semi-supervised algorithms with concatenated background information, as shown in Table V. ***ACVAE* outperforms the best-performing semi-supervised baselines by 39% to 289%, and also outperforms their enhanced variants by 3% to 42%,** which demonstrates the importance of utilizing background information and query model for semi-supervised MTS anomaly detection.

c) Supervised methods: As shown in Table III, ***ACVAE* with only 3% labels achieves similar or even better (-1.45% to +6.69%) performance than the best-performing supervised methods,** which demonstrates the effectiveness of the designs in *ACVAE* on utilizing out-of-band information to improve our detection performance.

Table IV shows the range-based AUPRC for *ACVAE* and baselines, and the anomaly detection performance on SKAB dataset. It clearly shows that *ACVAE* outperforms ALL baselines (including the supervised methods) on ALL the evaluated datasets, which highlights the effectiveness of *ACVAE* for situation-aware MTS anomaly detection.

2) RQ2. Incorporate Out-of-Band Information: We try to *enhance* the semi-supervised algorithms (Devnet, DeepSAD, and meta-AAD) with a strawman method to utilize the background information on CLM and GLM datasets. Specifically, we concatenate the background information with the raw MTS data as their input, and fine-tune their hyperparameters to reach the best performance (meta-AAD still uses 3% invalid test labels as it requires). The results are shown in Table V. Moreover, we show the performance of *ACVAE* without background information and *ACVAE*-concat, which use the strawman concatenate method to utilize background information (instead of using the learnable prior in *ACVAE*). The results show that, the semi-supervised methods can benefit from background information even with the strawman method, and **a proper design (learnable prior in *ACVAE*) to use the background information can further improve the performance by 7.42% to 9.21%**. Overall, ***ACVAE* outperforms the *best enhanced baselines* by 3% to 42%**.

TABLE IV

RANGE-BASED AUPRC FOR *ACVAE* AND BASELINES. THE LAST COLUMN (SK_F1) SHOWS THE BEST RANGE F1 SCORE ON SKAB DATASET. ALGORITHMS MARKED WITH * ARE THE *ENHANCED* BASELINES WITH CONCATENATED BACKGROUND INFORMATION. THE TOP-3 PERFORMANCE ON EACH DATASET ARE MARKED IN BOLD

Method	CLM	GLM	DCLM	SKAB	SK_F1
AutoEncoder	0.029	0.023	0.423	0.359	0.506
IsolationForest	0.012	0.016	0.154	0.381	0.516
USAD	0.029	0.028	0.653	0.383	0.631
InterFusion	0.042	0.065	**0.718**	0.407	0.551
Ymir 0%	0.071	0.022	0.267	0.359	0.530
UAE-GD	0.121	0.016	0.507	0.336	0.544
GDN	0.031	0.027	0.687	0.392	0.494
LSTM-NDT	0.004	0.037	0.073	0.322	0.411
Ymir 100%	**0.829**	**0.517**	**0.957**	0.384	**0.656**
TapNet	0.461	0.072	0.463	0.331	0.522
Devnet	0.006	0.055	0.352	0.368	0.522
DeepSAD	0.026	0.042	0.587	0.423	0.617
meta-AAD	0.448	0.099	0.647	0.421	0.588
Ymir 3%	0.089	0.036	0.480	0.388	**0.638**
Devnet*	0.272	0.117	0.352	0.368	0.522
DeepSAD*	0.359	0.417	0.587	**0.423**	0.617
meta-AAD*	**0.486**	**0.542**	0.647	**0.421**	0.588
ACVAE	**0.926**	**0.543**	**0.968**	**0.436**	**0.683**

TABLE V

BEST RANGE F1 SCORE OF *ACVAE* AND ITS VARIANT AND ***ENHANCED*** BASELINES WITH OR WITHOUT BACKGROUND INFORMATION

Method (label portion)	CLM		GLM	
	w/o	with	w/o	with
Devnet* (3%)	0.0327	0.4107	0.1596	0.6174
DeepSAD* (3%)	0.0483	0.5244	0.1177	0.5756
meta-AAD* (3%)	0.5375	0.6791	0.1748	0.6611
ACVAE-concat (3%)	0.7746	0.8825	0.2228	0.6338
ACVAE (3%)	0.7746	**0.9638**	0.2228	**0.6808**

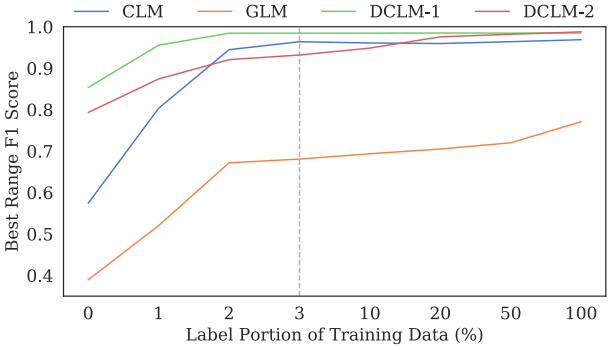


Fig. 4. Best range F1 score of *ACVAE* trained with different label portions of training data.

Fig. 4 shows the best range F1 score of *ACVAE* trained with different label portions of training data. Specifically, *ACVAE* with 0% labels is an unsupervised version of *ACVAE*, which only trains $\text{model}_{\text{full}}$ on MTS data and background information without feedback. **Incorporating 3% feedback labels help *ACVAE* outperform its unsupervised version by 15.35% to 74.87%**. *Even with only 1% labels, the performance of ACVAE significantly outperforms its unsupervised variant, which demonstrates the importance of incorporating feedback information to help the algorithm learn the operators' criteria about anomalies.*

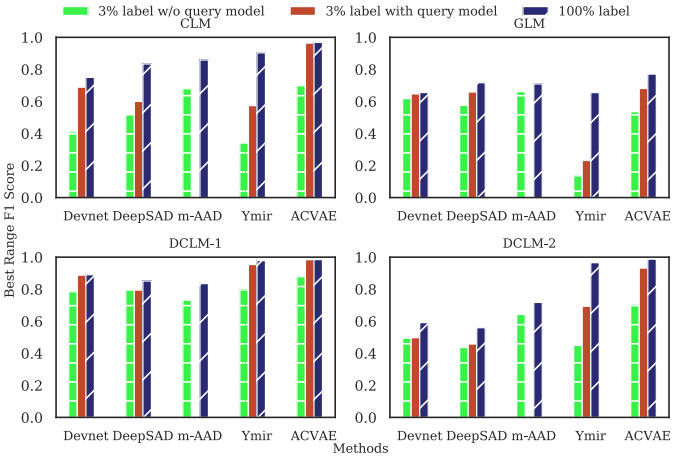


Fig. 5. Best range F1 score of *ACVAE* and ***enhanced*** baselines using *ACVAE*'s query model results.

3) RQ3. Query Model Performance: As shown in Fig. 4, *ACVAE* with only 2% to 3% labels already achieves comparable performance with the supervised version of *ACVAE*. It demonstrates that **our query model has found the most useful instances for improving our detection model with less than 3% queries of training data**, thus significantly reducing the amounts of labels required for training *ACVAE*.

Devnet, DeepSAD, and Ymir do NOT have a specific query model to proactively ask for labels. Based on the *enhanced* baselines in Table V, we further apply the *ACVAE*'s query model results (to select training labels) on these three methods to get new *enhanced variants* of baselines. (Note that, meta-AAD used its meta-policy for both query and detection, thus *ACVAE*'s query results cannot be applied on it.) The results are shown in Fig. 5. Although the query model of *ACVAE* is not jointly trained with other baselines, *its query results also improve the performance of other baselines up to 71%*, which shows that **the queried useful instances can also benefit other detection algorithms**. Moreover, with the same background information and feedback labels as input, ***ACVAE* still significantly outperforms the best-performing enhanced baselines by 2.98% to 39.90%**, which demonstrates the superiority of *ACVAE*'s designs on detection models to better utilize the out-of-band information. The semi-supervised *ACVAE* with 3% labels even outperforms many *enhanced* baselines trained with 100% labels.

F. RQ4. Ablation Studies

We conduct ablation studies using several variants of *ACVAE* on eBay's datasets mentioned in Section IV-A to demonstrate the effectiveness of the designs described in Section III.

1) Designs to Utilize Out-of-Band Information:

a) Incorporate background information: As shown in Table V, *ACVAE* with background information significantly outperforms *ACVAE* without background information by

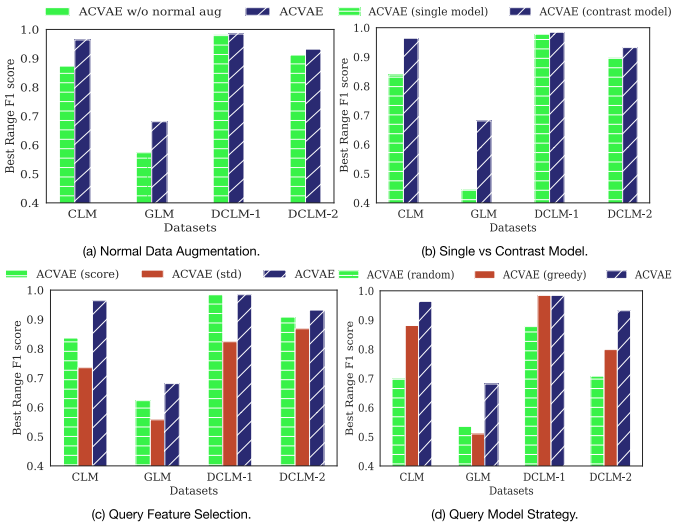


Fig. 6. Best range F1 score of *ACVAE* and its variants. (a) *ACVAE* with and w/o normal data augmentation. (b) *ACVAE* using contrast detection models and its variant using single detection model. (c) *ACVAE* using anomaly score and standard deviation of posterior as features for query model, as well as its variants using either of them. (d) *ACVAE* with neural network query model and its variants using random query and greedy query.

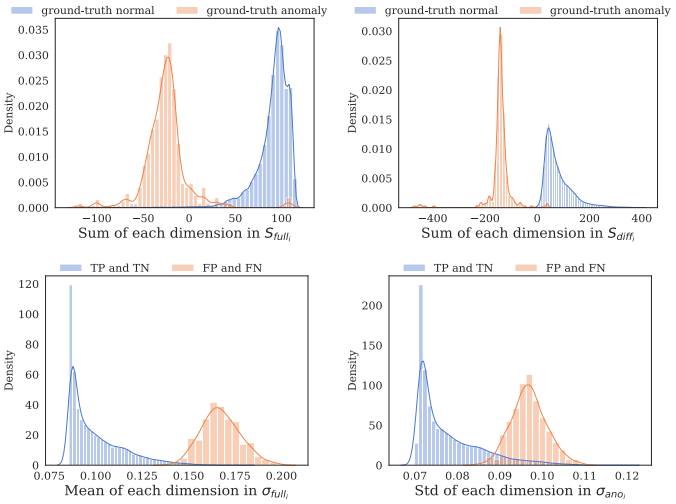


Fig. 7. Histograms of the statistics of four query features during training with respect to (1) ground-truth normal and anomalous instances and (2) correctly detected (TP and TN) and falsely detected (FP and FN) instances.

24.42% and 205.56% on CLM and GLM dataset, respectively. A similar trend can be found on other enhanced baseline methods, which demonstrates the importance of incorporating background information for situation-aware MTS anomaly detection.

b) Incorporate feedback information: As shown in Fig. 4, *ACVAE* with feedback information significantly outperforms the unsupervised variant of *ACVAE*. A similar trend can be found in Table III. Although previous works [3], [12], [15] have shown that unsupervised methods can achieve good performance on some MTS anomaly detection datasets through learning the normal patterns, they mainly focus on detecting data anomalies regardless of human operators' interests. Incorporating feedback information can help the anomaly detection algorithm learn the operators' criteria about anomalies, so as to make precise alerts in different kinds of monitoring systems.

c) Normal data augmentation: As shown in Fig. 6(a), *ACVAE* outperforms its variant without using normal data augmentation by 0.22% to 18.73%. Intuitively, for better anomaly detection with contrast detection models, we need $\mathrm{model}_{\mathrm{full}}$ and $\mathrm{model}_{\mathrm{ano}}$ to be well-trained on normal data and anomalous data, respectively. However, some normal instances are hard to learn without feedback, and $\mathrm{model}_{\mathrm{full}}$ will give low likelihood on such data (FPs). Fortunately, our query model is more likely to query such FP instances and get feedback labels. The normal data augmentation technique can then help $\mathrm{model}_{\mathrm{full}}$ optimize on such confirmed normal instances and get higher likelihood, which makes $\mathrm{model}_{\mathrm{full}}$ well-trained on normal data to meet the above requirements.

d) Contrast models vs single model: The variant model *ACVAE*-single removes the $\mathrm{model}_{\mathrm{ano}}$ in *ACVAE*, and only use $\mathrm{model}_{\mathrm{full}}$ to learn the normal patterns of data from raw MTS and out-of-band information. For queried anomalous instances, *ACVAE*-single removes them from its training data. The query model of *ACVAE*-single also removes the features generated by $\mathrm{model}_{\mathrm{ano}}$. *ACVAE* uses S_{diff} (Eq. (6)) as anomaly score, while *ACVAE*-single uses S_{full} (Eq. (4)). Other configurations are kept the same. The results in Fig. 6(b) shows that, *ACVAE* with contrast detection models outperforms *ACVAE*-single by 0.69% to 52.37%, which demonstrates the effectiveness of using contrast detection models and S_{diff} to learn both normal and anomalous patterns for MTS anomaly detection.

2) Designs of Query Model:

a) Query model feature selection: As shown in Fig. 7, we draw the histograms of the statistics of four query features during training procedure as examples: $(S_{full_i}, S_{diff_i}, \sigma_{full_i}, \sigma_{anoi})$ in Eq. (9)) with respect to (1) ground-truth normal and anomalous instances, (2) correctly detected (TP and TN) and falsely detected (FP and FN) instances. The figures show that most of the ground-truth anomalies can be distinguished by the score features, while the FP and FN instances can be distinguished by the standard deviation features. In Fig. 6(c), we compare *ACVAE* with two variant models. *ACVAE*-score only uses (S_{full_i}, S_{diff_i}) as query model features, and *ACVAE*-std only uses $(\sigma_{full_i}, \sigma_{anoi})$). *ACVAE* outperforms the best-performing variant by 6.81% on average over four datasets, which demonstrates the effectiveness of the selected features for training the query model.

b) Query model strategy: In Fig. 6(d), we replace *ACVAE*'s neural network based query model with random query and greedy query strategy. The random query is the same as the strategy in Devnet and DeepSAD, which randomly selects normal and anomalous data according to the anomaly ratio of each dataset to obtain feedback labels for training. The greedy strategy greedily selects the instances with the lowest S_{diff} (which means they are the most anomalous instances detected by the *current* detection model) after each week to obtain feedback labels for training. Other configurations are kept the same. *ACVAE* outperforms its variant models using random query and greedy query by 27.2% and 14.9% on average over four datasets, which demonstrates the effectiveness of

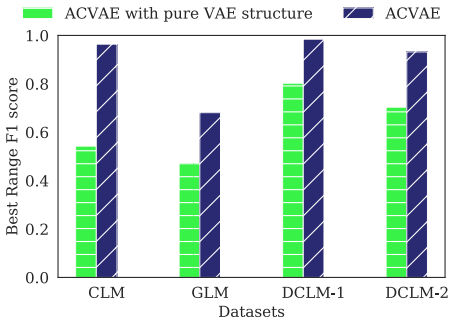


Fig. 8. Best range F1 of *ACVAE* and its variant model using a different detection model structure (pure VAE with multivariate unit Gaussian prior).

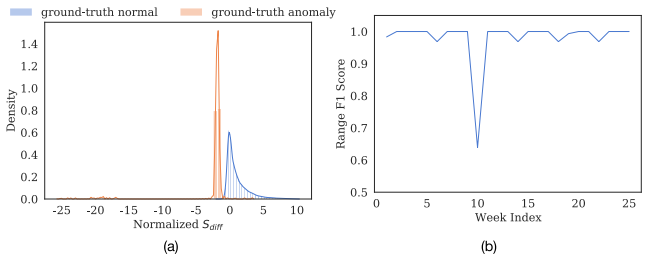


Fig. 9. (a) Histogram of normalized S_{diff} with respect to ground-truth normal and anomalous data on CLM dataset. (b) Online Training and testing performance on CLM dataset.

our query model strategy. Moreover, as shown in Table III and Table V, *ACVAE* significantly outperforms meta-AAD, which trains a model-specific query strategy using reinforcement learning. In Fig. 5, the query results of *ACVAE* can also help other semi-supervised methods (Devnet, DeepSAD, and Ymir) improve their anomaly detection performance, which shows the generalizability of our query results.

3) Designs of Detection Model Structure:

a) Detection model structure: In Fig. 8, we replace the state space model structure (which connects the latent variables $\mathbf{z}s$ in posterior and prior) in the detection model of *ACVAE* with pure VAE structure [18] and multivariate unit Gaussian prior. Other configurations are kept the same. *ACVAE* outperforms its variant model by 22.66% to 77.65%. With a simple VAE structure, the variant model is hard to model the temporal and correlational information of MTS, which prevents the contrast detection models from learning the normal and abnormal MTS patterns well. Moreover, the variant model cannot utilize the background information well without the learnable prior. These drawbacks significantly downgrade its performance.

V. Feasibility and Case Study

In this section, we use several statistics and production cases to show the feasibility of deploying *ACVAE* in production.

A. Feasibility Study

1) Anomaly Detection Performance: As discussed in Section IV-E, *ACVAE* with only 3% labels achieves comparable or even better (-1.45% to $+6.69\%$) anomaly detection performance than the supervised method Ymir deployed in production, which demonstrate *ACVAE*'s high accuracy and low label demand on anomaly detection for different kinds of systems. Moreover, the average anomaly detection delay of *ACVAE* is 1.269 points over the four datasets, which meets the early detect requirement in industry.

2) Computation Time: As shown in Fig. 2, *ACVAE* is incrementally trained once a week. The total training time ranges from 4 to 22 minutes per week on CLM, GLM, and DCLM dataset with a single NVIDIA GeForce GTX 2080 Ti graphical card. The online testing time for each instance is less than 0.02 seconds (much smaller than the data collection interval of 1 minute), which meets the real-time detection requirements.

3) Cold Start and Online Training: We take CLM as an example to show the feasibility for online training and detection. Even if there is only one anomaly case available at the beginning of training, the detection performance of *ACVAE* drops less than 3%. Moreover, besides training and testing with fixed data split in Section IV-E, we show the online result (*i.e.*, training the model online and testing on the following one week's data) in Fig. 9(b). It shows that *ACVAE* can achieve high online anomaly detection performance after training with one-week data. At week 9, *ACVAE* fails to detect one new anomaly that only lasts for 1 minute. However, *ACVAE* proactively queries for user feedback after that week and quickly learns the new anomalous pattern, and achieves best range F1 higher than 0.95 in the following weeks. During online training and testing, *ACVAE* dynamically calculates the threshold for obtaining the best F1 score on the past labeled instances, and use it for anomaly detection on the next week's data. On the 6-month CLM data, *ACVAE* successfully detects 18 out of 19 anomaly segments (more than 800 true positive anomaly points), with only 16 falsely alert points. Therefore, *ACVAE* has the cold start and incremental online training ability to be deployed on new systems.

4) Query Frequency: To deploy an active learning algorithm in production, it's often impossible to receive frequent feedback from operators and downstream users. *ACVAE* only queries once per week to get few user feedback, and converges to a good query policy with less than 5 episodes of queries. This is in contrast with the reinforcement learning based methods which often need more frequent feedback and more labels (*e.g.*, 300 episodes or more) to converge. Moreover, *ACVAE* is able to continuously learn new normal patterns from unlabeled data even if no feedback is provided for a few weeks. Furthermore, operators can proactively provide some feedback labels to further improve model training.

B. Case Study

We use several cases from different monitoring systems to show how *ACVAE* reduces the number of FPs and FNs by incorporating out-of-band information. To make the anomaly scores from different models comparable, we divide the original anomaly score by the average score across test data as the normalized anomaly score. Take CLM as an example, after

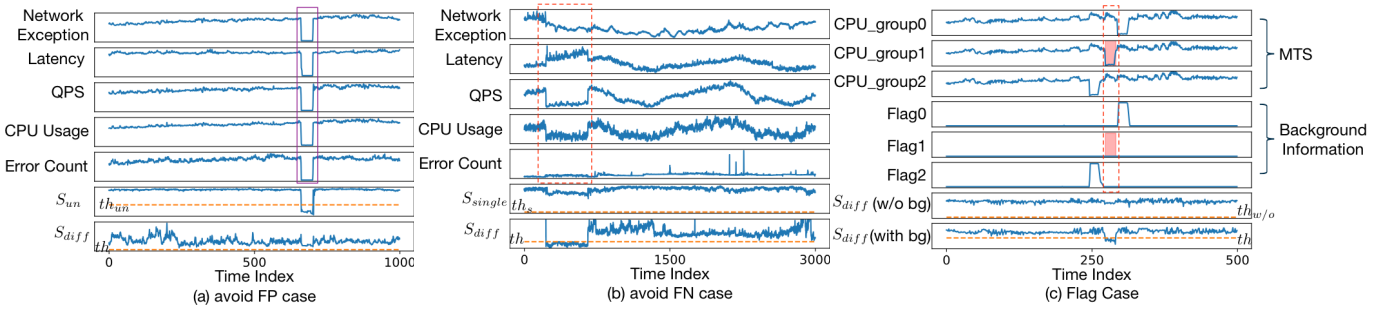


Fig. 10. Case Study on how *ACVAE* reduce the False Positives and False Negatives through incorporating out-of-band information. (a) avoid FP case. The purple strip is a ground-truth normal event for operators. (b) avoid FN case. The red strip is a ground-truth anomalous event for operators. (c) Flag case. The red strip is a ground-truth anomalous event for operators. "w/o bg" means *ACVAE* variant without background information. "with bg" means *ACVAE* with background information.

initialized with the well-trained $\text{model}_{\text{full}}$ and fine-tuned on labeled anomalous data, $\text{model}_{\text{ano}}$ learns the anomalous patterns and assigns 3.90 higher scores on average than $\text{model}_{\text{full}}$ for anomalous data. For normal data, $\text{model}_{\text{ano}}$ assigns similar or lower (-1.01 on average) score than $\text{model}_{\text{full}}$. In Fig. 9(a), $S_{diff} = -(S_{ano} - S_{full})$ for most anomalous data are smaller than S_{diff} for normal data (points with smaller scores are more anomalous), thus serve as an effective anomaly indicator.

1) Case 1: Detecting FP Case: Fig. 10(a) shows a service restart event in CLM dataset (the purple strip). Most metrics drop down for about 45 minutes, and here we show part of them. Such events are often launched by operators, and are regarded as normal events. However, since the service restart rarely happens, the unsupervised methods often falsely detect it as an anomaly (*i.e.*, False Positive). Here we take the unsupervised version of *ACVAE* as an example, the average anomaly score $S_{un} = -2.22$, smaller than its threshold $th_{un} = -1.37$, which means the unsupervised algorithm falsely detect this event as an anomaly. However, with the help of feedback information, *ACVAE* trains $\text{model}_{\text{full}}$ with normal data augmentation, and average $S_{full} = -0.95$ for this event. $\text{model}_{\text{ano}}$ only fine-tuned on anomalous patterns, thus gives similar score $S_{ano} = -0.43$ for the learned normal pattern. Finally, average $S_{diff} = -0.52$, higher than the threshold of *ACVAE* $th = -1.48$, thus detect this event as a normal one. Incorporating feedback information help *ACVAE* reduce the number of FPs through normal data augmentation and contrast detection models.

2) Case 2: Detecting FN Case: Fig. 10(b) shows a congestion anomaly event in CLM dataset (the red strip). The event mainly occurs in one module of the monitored cluster, causing the metrics associated with that module to be slightly lower than usual and last for about 7 hours, while other metrics perform as usual. The semi-supervised single-model variant *ACVAE*-single only learns the normal patterns, thus only detecting the significantly deviations as anomalies (with threshold $th_{single} = -5.84$). For the anomalous event in Fig. 10(b), *ACVAE*-single gives average score of $S_{single} = -1.68$, and falsely detects it as an normal event (*i.e.*, False Negative). For *ACVAE* with contrast detection models, the average $S_{full} = -1.87$. $\text{model}_{\text{ano}}$ is fine-tuned with similar anomalous events before according to user feedback, thus assigning average $S_{ano} = -0.02$, which is much higher than S_{full}. It means that $\text{model}_{\text{ano}}$ thinks this event is more similar with the anomalous patterns, leading to $S_{diff} = -1.85$, lower than the threshold of *ACVAE* $th = -1.48$. Therefore, this event is detected as an anomaly by *ACVAE*. Leveraging feedback information by contrast detection models makes *ACVAE* able to learn both normal and anomalous patterns, thus reducing the number of FNs compared with the single-model method that only learns the normal patterns.

3) Case 3: Detecting Flag Case: Fig. 10(c) shows a code deployment anomaly event in GLM dataset (the red strip). Groups 0, 1, 2 denote three out of seven groups of machines in a cluster. The operator wants to deploy code on groups 0, 2, 4, 6, thus the automated deployment system records deployment flags (*i.e.*, background information) in system logs and successively does the deployment. However, as shown in the red strip of Fig. 10(c), group 1 (which belongs to another data center and should not be deployed in this batch) also shows the deployment feature. In fact, it is caused by a configuration bug in the automated deployment system, and the system falsely deployed the same code on group 1. Without incorporating background information, the variant *ACVAE*-nobg regard the deployment on groups 0, 1, 2 as normal events according to the raw MTS. For the red strip, it assigns an average score $S = 0.03$, higher than its threshold $th_{w/o} = -2.78$, thus detecting it as a normal event. *ACVAE* incorporates the background information through a learnable prior, thus learning the criteria that groups with the same deployment status should follow similar correlations. Therefore, group 1 with $flag = 0$ and in-deployment CPU feature should be an anomaly. *ACVAE* assigns $S_{diff} = -0.82$ for the red strip, lower than its threshold $th = -0.50$ on GLM data, thus detects it as an anomalous event. With the help of background information, *ACVAE* is able to learn the system status to improve its anomaly detection performance.

C. Discussions

1) Background Information: System operators choose proper background information according to what extra information they need to detect anomalies in a specific system. The common background information can be encoded into 3 types: one-hot vectors, real-valued series, and no background

information (*i.e.*, the anomaly can be detected based on raw MTS data). In this paper, we evaluate *ACVAE* and baselines on three representative datasets, CLM (one-hot vector), GLM (real-valued series), and DCLM (no background information), each of which corresponding to one type of background information. These types of background information are sufficient to cover most monitoring scenarios in eBay. We believe other background information in new systems can be encoded in a similar way to be used in *ACVAE*.

2) Query Model: We stick to using a neural network model with proper feature selection as our query model, rather than using a reinforcement-learning (RL) based model [41]. The main reason is that the RL-based methods often need more episodes (*e.g.*, 300 episodes or more) and more labeled data to converge [41], [49], which means that we need to ask users for feedback frequently. However, this is often impractical in system operation scenarios. With the help of domain knowledge and proper feature selection, our query model can query once (20 data windows) per week and converge to a good query policy with less than 5 queries (episodes), which makes it more practical for MTS anomaly detection in industrial systems.

VI. Related Work

A. Unsupervised MTS Anomaly Detection

Anomaly Detection [10] has been widely studied for a decade of years. Specifically, MTS anomaly detection focusing on the multivariate time series data, such as system monitoring metrics [2], [3], [15], sensor data [5], [13], [14], [17], telemetry data [12], which are widely used in manufacturing industry and Information Technology systems. Recent years, many unsupervised deep learning based MTS anomaly detection methods [2], [3], [5], [12]–[17], [19] have been proposed, which aim at detecting "data anomaly" from the raw MTS data under proper assumptions. In general, these unsupervised methods have a common underlying assumption, *i.e.*, the "normal patterns" of data are generated from a deterministic procedure, and it's able to learn the distribution (or prediction) of "normal patterns" from raw MTS data [42], [50]. Data instances that deviate from the learned normal distribution (or prediction) are regarded as anomalies. In this way, these methods mainly detect the "data anomaly", which is directly defined by the historical MTS data. Detailedly, [12], [24] used LSTM-based [51] method and multi-head attention [52] based methods for MTS prediction. [5], [19] used an Encoder-Decoder structure to learn MTS reconstructions. [13], [15] incorporated adversarial training to model the inter-metric dependency (*i.e.*, correlations among metrics). [2], [3], [14] used VAE-based [18] methods to model the normal patterns of MTS from temporal or correlational perspectives. Finally, the prediction error or reconstruction error (which represents how far a data instance deviates from the learned normal patterns) is used for anomaly detection. The unsupervised methods often focus on systems with *stable* services and *periodical* monitoring data, where the anomalies are mainly "data anomaly" that can be directly learned from historical MTS data (NOT *situation-aware*).

B. Semi-Supervised and Active Anomaly Detection

In a broader range of anomaly detection, real anomalies are defined not only by historical data, but also relate to domain experts' criteria for anomalies [42], [53]–[55]. Therefore, it's important to use semi-supervised approaches to learn domain experts' criteria for MTS anomalies from out-of-band information.

Several semi-supervised algorithms have been developed for anomaly detection, mainly applied on images, graphs, and manually extracted features [42], [43], [54], [56], [57]. [56], [57] used a belief propagation process to utilize labeled anomalies, but are only applicable to graph data. [54] leveraged a few labeled anomalies to improve the learned feature representations for different applications. Devnet [42] proposed a neural deviation network to directly optimize the anomaly scores with a few labeled anomalies and a Gaussian prior on anomaly reference score. DeepSAD [43] generalized Deep SVDD [58] and leveraged an information-theoretic idea to split normal and anomalous data in a learned latent space from labeled normal and anomalous data. These methods are developed for images or graph data, thus cannot leverage the background information for MTS data. Moreover, these methods need a number of labeled anomalies for training, but do not have a query model to help them find such instances from unlabeled data. In MTS data, anomalies are often rare, which makes it expensive to inspect large amounts of unlabeled data and find sufficient anomalies for training.

As a special case of semi-supervised anomaly detection, active anomaly detection methods leverage the active learning idea on unbalanced anomaly detection data to find the most useful instances for optimizing the detection model. [53], [59], [60] combines different detection models with a greedy strategy to query the most anomalous instance according to anomaly score and get feedback label. [49] applied margin sampling with label propagation as the query strategy to train a DQN detector. Meta-AAD [41] trained a meta-policy with PPO (a reinforcement learning method) as its query strategy, which models the long-term performance for querying the useful instances and outperforms the above strategies.

VII. Conclusion

In this paper, we point out the importance of incorporating out-of-band information (including background information and feedback information) to learn domain experts' criteria about anomalies for situation-aware MTS anomaly detection in large distributed systems. We first propose an end-to-end algorithm *ACVAE* that incorporates the out-of-band information into MTS anomaly detection through active learning and contrast VAE-based models. Specifically, we propose a pair of contrast detection models to learn both normal and abnormal patterns from user feedback, as well as a conditional VAE with learnable prior to modeling the background information. We adopt a non-linear state space model structure to capture the temporal and correlational information of MTS data to better learn normal and abnormal patterns. We also propose a novel query model for VAE-based anomaly detection methods, which proactively queries the most useful instances for user

feedback. We evaluate *ACVAE* on four datasets collected from different monitoring scenarios in search backend systems at eBay, as well as one public sensor dataset. *ACVAE* with only 3% labels significantly outperforms the state-of-the-art unsupervised and semi-supervised MTS anomaly detection methods. Moreover, *ACVAE* even achieves the performance comparable to or better than the supervised method developed by domain experts at eBay, which demonstrates the feasibility of deploying *ACVAE* in production. The effectiveness of each design in *ACVAE* has been verified through ablation studies. We also use several statistics and production cases to show the feasibility of applying *ACVAE* in production.

For future work, we would like to incorporate more complex out-of-band information (*e.g.*, semantic information that summarized domain experts' knowledge about anomalies, topological information about underlying system structure) to learn domain experts' criteria for situation-aware anomaly detection in communications and networks.

ACKNOWLEDGMENT

The authors thank the anonymous reviewers for the valuable feedback.

REFERENCES

[1] G. Y. Li *et al.*, "Series editorial: Inauguration issue of the series on machine learning in communications and networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 1–3, Jan. 2021.

[2] 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 *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2828–2837.

[3] Z. Li *et al.*, "Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 3220–3230.

[4] T. Li, Y. Geng, and H. Jiang, "Anomaly detection on seasonal metrics via robust time series decomposition," in *Proc. 1st Workshop Int. Joint Conf. Artif. Intell. (AI4AN)*, 2020, pp. 1–6.

[5] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based encoder–decoder for multi-sensor anomaly detection," in *Proc. Int. Conf. Mach. Learn. Anomaly Detection Workshop*, 2016, pp. 1–5.

[6] H. Ren *et al.*, "Time-series anomaly detection service at Microsoft," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 3009–3017.

[7] H. Xu *et al.*, "Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications," in *Proc. World Wide Web Conf. (WWW)*, 2018, pp. 187–196.

[8] Z. Chen, D. Chen, X. Zhang, Z. Yuan, and X. Cheng, "Learning graph structures with transformer for multivariate time series anomaly detection in IoT," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9179–9189, Jun. 2021.

[9] O. Salem, K. Alsubhi, A. Mehaoua, and R. Boutaba, "Markov models for anomaly detection in wireless body area networks for secure health monitoring," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 2, pp. 526–540, Feb. 2021.

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

[11] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices architecture enables DevOps: Migration to a cloud-native architecture," *IEEE Softw.*, vol. 33, no. 3, pp. 42–52, May /Jun. 2016.

[12] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 387–395.

[13] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks," in *Proc. Int. Conf. Artif. Neural Netw.* Cham, Switzerland: Springer, 2019, pp. 703–716.

[14] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1544–1551, Jul. 2018.

[15] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "USAD: Unsupervised anomaly detection on multivariate time series," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 3395–3404.

[16] A. Garg, W. Zhang, J. Samaran, R. Savitha, and C.-S. Foo, "An evaluation of anomaly detection and diagnosis in multivariate time series," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 6, pp. 2508–2517, Jun. 2022.

[17] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 5, pp. 4027–4035.

[18] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. 2nd Int. Conf. Learn. Represent. (ICLR)*, 2014, pp. 1–14.

[19] C. Zhang *et al.*, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 1409–1416.

[20] N. Zhao *et al.*, "Automatic and generic periodicity adaptation for KPI anomaly detection," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 3, pp. 1170–1183, Sep. 2019.

[21] Y. Ikeda, K. Tajiri, Y. Nakano, K. Watanabe, and K. Ishibashi, "Estimation of dimensions contributing to detected anomalies with variational autoencoders," in *Proc. AAAI Conf. Artif. Intell. Netw. Interp. Deep Learn. Workshop*, 2019, pp. 1–8.

[22] B. Settles, "Active learning literature survey," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, Tech. Rep. 1648, 2009.

[23] C. C. Aggarwal, X. Kong, Q. Gu, J. Han, and S. Y. Philip, "Active learning: A survey," in *Data Classification: Algorithms and Applications*. Boca Raton, FL, USA: CRC Press, 2014, pp. 571–605.

[24] S. Huang, D. Wang, X. Wu, and A. Tang, "DSANet: Dual self-attention network for multivariate time series forecasting," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2019, pp. 2129–2132.

[25] Z. Xiao, Q. Yan, and Y. Amit, "Likelihood regret: An out-of-distribution detection score for variational auto-encoder," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 20685–20696.

[26] W. Chen, X. Nie, M. Li, and D. Pei, "DOI: Divergence-based out-of-distribution indicators via deep generative models," 2021, *arXiv:2108.05509*.

[27] S. Roweis and Z. Ghahramani, "A unifying review of linear Gaussian models," *Neural Comput.*, vol. 11, no. 2, pp. 305–345, Feb. 1999.

[28] M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther, "Sequential neural models with stochastic layers," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2199–2207.

[29] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proc. 31st Int. Conf. Mach. Learn.*, vol. 32, 2014, pp. 1278–1286.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[31] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," 2018, *arXiv:1811.03378*.

[32] *Binary Crossentropy*. Accessed: Nov. 18, 2021. [Online]. Available: https://www.tensorflow.org/versions/r1.15/api_docs/python/tf/keras/back%end/binary_crossentropy

[33] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[34] *Elastic Beats*. Accessed: Mar. 08, 2022. [Online]. Available: https://www.elastic.co/cn/beats/metricbeat

[35] *Prometheus*. Accessed: Mar. 08, 2022. [Online]. Available: https://prometheus.io/

[36] I. D. Katser and V. O. Kozitsin. (2020). *Skoltech Anomaly Benchmark (SKAB)*. [Online]. Available: https://www.kaggle.com/dsv/1693952

[37] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proc. 2nd Workshop Mach. Learn. Sensory Data Anal. (MLSDA)*, 2014, pp. 4–11.

[38] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining*, Dec. 2008, pp. 413–422.

[39] Z. Zhao, "Ymir: A supervised ensemble framework for multivariate time series anomaly detection," 2021, *arXiv:2112.04704*.

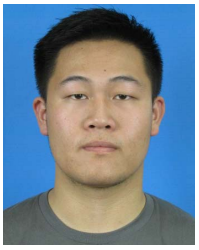
[40] X. Zhang, Y. Gao, J. Lin, and C.-T. Lu, "TapNet: Multivariate time series classification with attentional prototypical network," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 6845–6852.

[41] D. Zha, K.-H. Lai, M. Wan, and X. Hu, "Meta-AAD: Active anomaly detection with deep reinforcement learning," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2020, pp. 771–780.
[42] G. Pang, C. Shen, and A. van den Hengel, "Deep anomaly detection with deviation networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 353–362.
[43] L. Ruff *et al.*, "Deep semi-supervised anomaly detection," in *Proc. 8th Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–13.
[44] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[45] B. G. Amidan, T. A. Ferryman, and S. K. Cooley, "Data outlier detection using the Chebyshev theorem," in *Proc. IEEE Aerosp. Conf.*, Mar. 2005, pp. 3814–3819.
[46] A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet, "Anomaly detection in streams with extreme value theory," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 1067–1075.
[47] N. Tatbul, T. J. Lee, S. Zdonik, M. Alam, and J. Gottschlich, "Precision and recall for time series," in *Proc. Int. Conf. Artif. Neural Netw.*, vol. 31, 2018, pp. 1920–1930.
[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2014, pp. 1–15.
[49] T. Wu and J. Ortiz, "RLAD: Time series anomaly detection through reinforcement learning and active learning," 2021, *arXiv:2104.00543*.
[50] D. M. Hawkins, *Identification of Outliers*, vol. 11. Dordrecht, The Netherlands: Springer, 1980.
[51] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[52] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[53] M. A. Siddiqui, A. Fern, T. G. Dietterich, R. Wright, A. Theriault, and D. W. Archer, "Feedback-guided anomaly discovery via online optimization," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2200–2209.
[54] G. Pang, L. Cao, L. Chen, and H. Liu, "Learning representations of ultrahigh-dimensional data for random distance-based outlier detection," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2041–2050.
[55] J. Gao, H. Cheng, and P.-N. Tan, "Semi-supervised outlier detection," in *Proc. ACM Symp. Appl. Comput. (SAC)*, 2006, pp. 635–636.
[56] M. McGlohon, S. Bay, M. G. Anderle, D. M. Steier, and C. Faloutsos, "SNARE: A link analytic system for graph labeling and risk detection," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 1265–1274.
[57] A. Tamersoy, K. Roundy, and D. H. Chau, "Guilt by association: Large scale malware detection by mining file-relation graphs," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 1524–1533.
[58] L. Ruff *et al.*, "Deep one-class classification," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 4393–4402.
[59] S. Das, W.-K. Wong, A. Fern, T. G. Dietterich, and M. A. Siddiqui, "Incorporating feedback into tree-based anomaly detection," 2017, *arXiv:1708.09441*.
[60] S. Das, W.-K. Wong, T. Dietterich, A. Fern, and A. Emmott, "Incorporating expert feedback into active anomaly discovery," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 853–858.

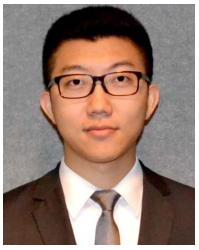
Zhihan Li (Student Member, IEEE) received the B.S. degree from Xidian University, Xi'an, China, in 2017. He is currently pursuing the Ph.D. degree with the Computer Science and Technology Department, Tsinghua University, Beijing, China. His current research interests include time series anomaly detection, and machine learning and its applications in network management.

Youjian Zhao received the B.S. degree from Tsinghua University, Beijing, China, in 1991, the M.S. degree from the Shenyang Institute of Computing Technology, Chinese Academy of Sciences, in 1995, and the Ph.D. degree in computer science from Northeastern University, China, in 1999. He is currently a Professor with the Computer Science and Technology Department, Tsinghua University. His research interests include high-speed Internet architecture, switching and routing, and anomaly detection for network data.

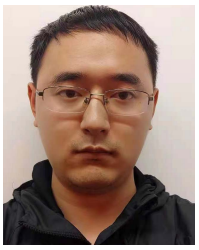
Yitong Geng received the master's degree from the University of Science and Technology of China in 2017. He is currently a Senior Algorithm Engineer with eBay Inc., focusing on improving the performance and reliability of the applications in the internet via state-of-the-art artificial intelligence techniques.



Zhanxiang Zhao received the master's degree from the University of Science and Technology of China in 2020. He is currently an Algorithm Engineer with eBay Inc. His work is about anomaly detection of system applications and improving monitoring efficiency via machine learning.



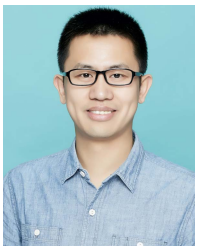
Hanzhang Wang received the Ph.D. degree in computing science from the University of Michigan. He joined eBay, as an Applied Researcher, in 2018. He also works as the Company-Wide University Partnership Program Manager. He is leading a team working on AIOps and AI4SE solutions. His recent research interests include intelligent observability, AIOps, RCA, SE, ML, and graph algorithms. He has multiple publications in top venues, including FSE, ASE, VLDB, TSC, and CIKM. The research outcomes have been transferred into multiple production products for anomaly detection, root cause analysis, and developer velocity.



Wenxiao Chen received the B.S. degree from Tsinghua University, Beijing, China, in 2017, where he is currently pursuing the Ph.D. degree with the Computer Science and Technology Department. His current research interests include time-series anomaly detection, machine learning, deep generative model, graph neural networks, root cause analysis, and out-of-distribution detection.

Huai Jiang joined eBay in 2011 and worked with eBay private cloud solution for six years. Since 2017, he has been working with eBay monitoring platform. He is currently the Principle MTS and an Architect at eBay. Prior to eBay, he worked in various areas, including networks and game for ten years.

Amber Vaidya is currently the Director of engineering with the Observability Platform Team, eBay. His team is responsible for building the centralized observability platform. This platform ingests and processes telemetry signals like metrics, logs, and traces across the entire eBay stack and allows for alerting and intelligent correlation.



Liangfei Su joined eBay in 2011. Since then, he has been working in multiple areas, including cloud, big data, and monitoring. He has engineering interests in the database management system and data analytic, and applying ML into engineering areas.

Dan Pei (Senior Member, IEEE) received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, in 1997 and 2000, respectively, and the Ph.D. degree from the University of California at Los Angeles, Los Angeles, CA, USA, in 2005. He is currently an Associate Professor with the Computer Science and Technology Department, Tsinghua University. His current research interests include anomaly detection and root cause analysis. Right now, he is focusing on the field of AIOps, which is at the intersection of AI, business, and IT operations.