Efficient KPI Anomaly Detection Through Transfer Learning for Large-Scale Web Services

Shenglin Zhang, *Member, IEEE*, Zhenyu Zhong, Dongwen Li, Qiliang Fan, Yongqian Sun, *Member, IEEE*, Man Zhu, Yuzhi Zhang, Dan Pei, *Senior Member, IEEE*, Jiyan Sun, Yinlong Liu, Hui Yang, and Yongqiang Zou

***Abstract*— Timely anomaly detection of key performance indicators (KPIs), *e.g.*, service response time, error rate, is of utmost importance to Web services. Over the years, many unsupervised deep learning-based anomaly detection approaches have been proposed. To achieve good performance, they require a long period of KPI data for model training, which is not easy to guarantee with frequent service changes. Additionally, the training overhead is too significant for the vast number of KPIs in large-scale Web services. To address the problems, we propose an unsupervised KPI anomaly detection approach, named *AnoTransfer*, by combining a novel Variational Auto-Encoder (VAE)-based KPI clustering algorithm with an adaptive transfer learning strategy. Extensive evaluation experiments using real-world data collected from several large-scale Web service providers demonstrate that *AnoTransfer* reduces the average initialization time by 65.71% and improves the training efficiency by 50.62 times, without significantly degrading anomaly detection accuracy.**

***Index Terms*— Key performance indicator, anomaly detection, time series clustering, transfer learning.**

Manuscript received 15 December 2021; revised 11 March 2022; accepted 5 May 2022. Date of publication 8 June 2022; date of current version 18 July 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB0300104, in part by the National Science Foundation of Tianjin under Grant 21JCQNJC00180, in part by the National Natural Science Foundation of China under Grant 61902200 and Grant 62072264, in part by the China Postdoctoral Science Foundation under Grant 2019M651015, and in part by the Beijing National Research Center for Information Science and Technology (BNRist). *(Corresponding author: Yongqian Sun.)*

Shenglin Zhang and Yuzhi Zhang are with the College of Software, Nankai University, Tianjin 300350, China, and also with the Haihe Laboratory of Information Technology Application Innovation and the Tianjin Key Laboratory of Operating System, Nankai University, Tianjin 300071, China (e-mail: zhangsl@nankai.edu.cn; zyz@nankai.edu.cn).

Zhenyu Zhong, Dongwen Li, Qiliang Fan, Yongqian Sun, and Man Zhu are with Nankai University, Tianjin 300071, China (e-mail: zyzhong@mail.nankai.edu.cn; lidongwen@mail.nankai.edu.cn; fanqiliang@mail.nankai.edu.cn; sunyongqian@nankai.edu.cn; zhuman2019@mail.nankai.edu.cn).

Dan Pei is with the Department of Computer Science, Tsinghua University, Beijing 100084, China, and also with the Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: peidan@tsinghua.edu.cn).

Jiyan Sun is with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100864, China (e-mail: sunjiyan@iie.ac.cn).

Yinlong Liu is with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100864, China, and also with the School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100864, China (e-mail: liuyinlong@iie.ac.cn).

Hui Yang and Yongqiang Zou are with Accumulus Technologies (Tianjin) Company Ltd., Tianjin 300392, China (e-mail: hui.yang@yunzhanghu.com; yongqiang.zou@yunzhanghu.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2022.3180785.

Digital Object Identifier 10.1109/JSAC.2022.3180785

I. Introduction

WEB services such as social networking, search engine, and online shopping have witnessed great success in recent years. These services produce a massive number of monitored key performance indicators (KPIs) in the form of time series, including system metrics (*e.g.*, CPU utilization, memory utilization, network throughput), user-perceived metrics (*e.g.*, service response time, error rate), and service performance (*e.g.*, page view counts). To timely catch the latent bugs or performance issues of Web services, anomaly detection on these KPIs is of great importance [1]–[13].

The monitoring data of KPI generally form two types of time series: the univariate time series (UTS) of a specific KPI and the multivariate time series (MTS) of an entity (*e.g.*, a physical machine, microservice, switch, router, docker, disk). Although MTS can give a more comprehensive picture of the monitored entity and MTS anomaly detection has gained much attention recently [9], [10], [14]–[18], detecting anomalies in UTS is still essential in modern operation tasks. Typically, operators are especially interested in the UTS of core KPIs, and conducting MTS anomaly detection cannot fully capture the anomalous behavior of these core KPIs. For example, operators usually focus on three core KPIs of a microservice, *i.e.*, average response time, error rate, and page view count. The MTS anomaly detection covering dozens of KPIs tends to miss the three core KPIs' anomalous behaviors or report many alarms that operators do not care about. Therefore, in this paper, we mainly focus on detecting anomalies for the UTS of KPIs (from now on, we use "KPI" to denote the UTS of a KPI), as shown in Figure 1.

However, since current KPI anomaly detection algorithms are mostly based on deep neural networks, it takes a long initialization time[1] (up to weeks) to achieve the desired performance. Another drawback of current algorithms is that the overhead of training a specific model for each monitored KPI is usually too high because there can be hundreds of thousands of KPIs in large-scale Web services. Intuitively, we can train a few deep learning-based KPI anomaly detection models based on a small number of KPIs and "transfer" the learned models to all other KPIs. In this way, we can shorten the model initialization time for a newly updated entity's KPI and reduce

[1]The initialization time of an anomaly detection method is the time lag between when it is deployed and when it becomes effective, mainly influenced by the length of historical data they need to digest.

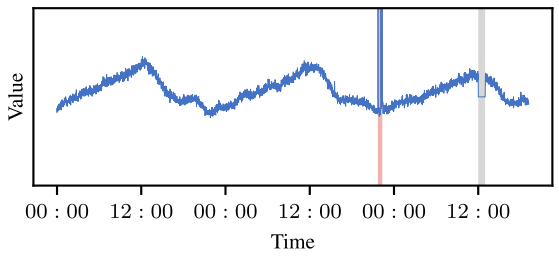

Fig. 1. A toy example of KPI. The red vertical span (the left one) denotes anomaly points, and the gray vertical span (the right one) denotes missing points.

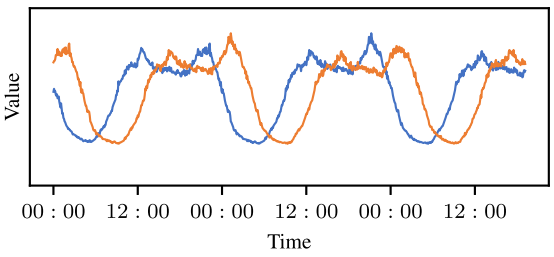

Fig. 3. An example of KPI phase shifts: two KPIs are similar in shape but have a time lag.

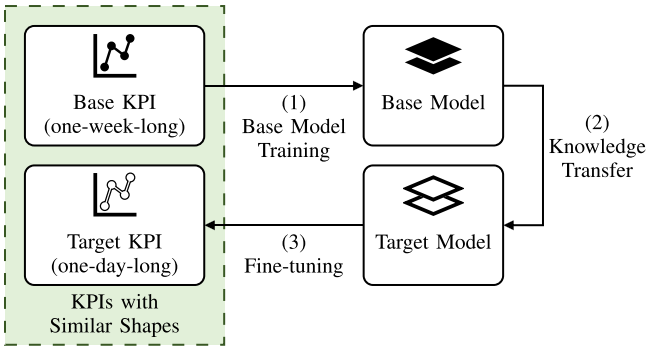

Fig. 2. The core idea of transfer learning for KPI anomaly detection.

the training overhead for large-scale Web services. Therefore, transfer learning, which applies the knowledge gained while training a model for one long-period KPI to another KPI (maybe with a short period) with similar patterns (as shown in Figure 2) is promising to solve our problem. We denote KPIs in the source domain of transfer learning as *base KPIs* and those in the target domain as *target KPIs*. Nevertheless, applying transfer learning to solve the above problem faces the following three challenges:

1) **High diversity of KPIs..** Heterogeneous entities can produce KPIs with different patterns. Typically, the model trained through a base KPI does not apply to a target KPI with very different patterns. Accordingly, matching a target KPI to a randomly selected base KPI is likely to suffer from a low anomaly detection accuracy (see § IV-D). Moreover, there are many situations where two KPIs are similar in shape but have a phase shift, as shown in Figure 3. Phase shifts further increase the diversity of KPI patterns.
2) **Short-period data for matching a target KPI to a base KPI.** It is vital to match a target KPI to a base KPI with a similar pattern. However, a target KPI may only have a short period of data because of software/hardware change or service instance creation. It brings a significant challenge to KPI matching because previous methods such as [19] require a long period of target KPI data (1 month), which is the same as base KPI, to find matched base KPIs. Moreover, long-period KPIs are likely to contain anomalies or pattern changes, making it even harder to determine which base KPI suits the target KPI best.
3) **The selection of transfer strategy and anomaly score threshold.** There are various strategies for transferring knowledge between models. The distances between the base and target KPI are various, rendering a fixed strategy inappropriate in our scenario. Adaptively choosing the best transfer strategy is needed to improve transfer learning performance. Additionally, operators have to manually set the anomaly score threshold for each KPI with previous methods like [1], [2], which is very tedious and error-prone. A more practical way to predict the anomaly score threshold on the fly is required.

We propose *AnoTransfer*, an efficient, unsupervised, transfer learning-based framework for KPI anomaly detection, to tackle the above challenges. First, it splits offline training data into one-day-long segments and categorizes them into groups using a novel shape-based clustering method. Then, *AnoTransfer* selects the KPI segments closest to each group's centroid to train a base model for this group jointly. When a new KPI arrives in the online environment, we match it to a group and fine-tune the group's base model through the KPI's short-period data with an adaptive transfer strategy. The contributions of this paper are summarized as follows:

1) To reduce the required initialization time and the training cost for large-scale Web services, we apply transfer learning to transfer the knowledge from well-trained base models to target models. We identify the challenges lying in addressing this problem.
2) We design a novel VAE-based KPI clustering algorithm for deep learning-based anomaly detection models. Through clustering, KPI segments with similar shapes can jointly train a base model. Using KPIs with similar shapes in transfer learning improves the performance of anomaly detection models, addressing the first challenge. We split KPIs by days and perform clustering on these one-day-long KPI segments, making it easy to assign base models for target KPIs, addressing the second challenge. Moreover, using shape-based distance (SBD) [20], we successfully minimize the phase shifts in KPIs, making the number of clusters and the number of base models as small as possible.
3) We propose a transfer learning strategy selection method to adaptively choose between full parameter transfer and partial parameter transfer according to the distance between the centroid of the base KPIs and the new KPI. SPOT [21] is also used with transfer learning techniques

to improve its performance in calculating the anomaly score threshold, addressing the third challenge.

4) We conduct extensive experiments with real-world data collected from large Web service providers including Baidu, Sogou, eBay, Tencent, and ByteDance to evaluate the performance *AnoTransfer*. Empirical results show that *AnoTransfer* reduces the average initialization time by 65.71%, and improves the training efficiency by 50.62 times, without significantly degrading the anomaly detection F1-score. Additionally, to get readers better understand our work, we also make our labeled dataset and source code publicly available.

II. Background and Related Work

A. KPI and KPI Anomaly Detection

KPIs are primarily affected by user behavior and schedule; therefore, most of them have seasonal patterns [2]. However, because users' behavior may change over time, the shapes of KPI curves are not the same in each repeated cycle. Besides seasonal patterns and local variations, there are also noise factors in KPIs. The statistical information of the noise distribution, *e.g.*, the standard deviation, is mainly discussed in KPI anomaly detection problems [1]. In summary, the *normal patterns* of seasonal KPIs consist of two components: 1) the seasonal patterns with local variations, and 2) the distribution of noises. In UTS, we refer to data points that do not follow normal patterns as *anomaly points* (such as the steeply rising or falling parts of a KPI). However, "normal patterns" of a variable in MTS can still be anomalous if other variables contain anomalous patterns at that time.

The KPIs discussed and studied in this paper are in the form of UTS. The anomaly detection problem of KPIs can now be formulated as follows: For each time t, given its KPI observation x_t and historical observations $\mathbf{x} = (x_{t-W+1}, \ldots, x_{t-1})$ of length W, determine whether an anomaly has occurred (denoted by $\gamma_t = 1$). Usually, anomaly detectors will give a real-valued score (*anomaly score* hereafter), *e.g.*, $P(\gamma_t = 1 \mid x_{t-W+1:t})$ instead of directly determining whether an anomaly has occurred at time t. Then, an anomaly score threshold is selected to determine whether to trigger alerts.

B. Service Update

With the current rapid evolution of the Internet, service updates are inevitable and frequent in large-scale Web services [4], [22]–[24]. These updates aim to fix bugs, deploy new features, adapt to environmental change, and improve service performance continuously. Although each service update is usually extensively tested on testbeds before deployment, errors and bugs may still occur in the online environment because of diverse hardware/software systems, complex interactions, and the large scale of devices [22], [23]. Therefore, instead of rolling out a service update to all servers at one time, the operations team deploys the service update on a subset of servers at the beginning and continuously monitors a predefined list of KPIs to determine the impact of the service update. If the KPIs on the server subset perform as expected, the service update will be rolled out to all servers. Otherwise, the service update should be rolled back as soon as possible to mitigate its negative impact.

Generally, service performance degradation caused by bad service updates may induce poor user experience or revenue drop [22], [23]. Thus it is vital to detect significant KPI changes rapidly, whether positive or negative, to allow a timely decision. For example, if operators replace an old server with a new high-performance server, a decrease could incur in CPU usage and response time. Apparently, this is an expected KPI change. Suppose that operators use [2] for anomaly detection, and the parameters are trained based on the old KPI shape (seasonal patterns with local variations) before this service update. Since the data distribution has changed dramatically, [2] will generate a lot of false alarms. That is to say, the anomaly detector "believes" that most of the KPI data points after the service update are anomalous, while operators consider these points as normal.

We try to adapt anomaly detection methods as quickly as possible to avoid accuracy loss after expected KPI changes. The requirement to update and redeploy the anomaly detection system timely then leaves us limited time to collect historical data for changed KPIs after service updates.

C. Conditional Variational Autoencoder

Variational autoencoder (VAE) [25] is a typical deep Bayesian network. It models the relationship between the input $\mathbf{x}$ and the latent variable $\mathbf{z}$. Like other autoencoders, a VAE model consists of an encoder and a decoder. Encoders and decoders are both neural networks, usually with similar structures. The task of the encoder is to compress the input information into a constrained latent distribution (encode). The task of the decoder is to sample latent variables from the latent space and then transform them back to high-dimensional data accurately (reconstruction). The training goal of VAE is to minimize the distribution distances between the estimated posterior $q(\mathbf{z} \mid \mathbf{x})$ and the real one $p(\mathbf{z} \mid \mathbf{x})$. This can be done by maximizing the log-likelihood of the reconstructed data $\mathbb{E}[\log p(\mathbf{x} \mid \mathbf{z})]$ to improve its quality while minimize the Kullback-Leibler divergence [26] between the posterior $q(\mathbf{z} \mid \mathbf{x})$ and the prior $p(\mathbf{z})$.

Conditional variational autoencoder (CVAE) [27] combines conditional generative models [28] with VAE to get a stronger control of what kind of data it generates. The encoder part of CVAE tries to learn $q(\mathbf{z} \mid \mathbf{x}, \mathbf{y})$, which is equivalent to encoding $\mathbf{x}$ into the hidden representation $\mathbf{z}$ conditioned on $\mathbf{y}$. The decoder part of CVAE tries to learn $p(\mathbf{x} \mid \mathbf{z}, \mathbf{y})$, which decodes the hidden representation to the input space conditioned on $\mathbf{y}$. Previously methods such as [2] used CVAE for anomaly detection, which demonstrated the superior performance of CVAE in real-world cases. Therefore, *AnoTransfer* adopts CVAE to learn the normal patterns of KPIs. Then, it compares the KPI reconstructed by the decoder with the input KPI to determine anomalous points.

D. Transfer Learning

Transfer learning focuses on storing knowledge gained while solving one problem and applying it to a different yet related problem [29]. There are different transfer learning

strategies and methods, which can be applied based on the domain, task, and availability of data [30], [31]. These methods can be classified into instance transfer, feature-representation transfer, parameter transfer, and relational-knowledge transfer. Instance transfer reuses certain instances from the source domain along with target data to improve the results of the target task. Feature-representation transfer aims to minimize domain divergence by discovering good feature representations that can be utilized from the source to target domains. Parameter transfer assumes that the models for related tasks share some parameters or prior distributions. Relational-knowledge transfer attempts to handle data that is not independent and identically distributed. When using transfer learning techniques, many deep generative models (*e.g.*, [32]) adopt parameter transfer because they often share the same prior distribution across the source and the target domain. Using parameter transfer to warm up the training process is closely related to Meta-learning based model initialization/selection, which picks a model for a new task based on the task similarity to the meta-train tasks [33]. The transfer strategies that *AnoTransfer* uses also fall into this category.

E. Related Work

1) KPI Clustering: **KPI clustering based on traditional methods.** ROCKA [19] and SPF [34] are two KPI clustering methods based on traditional statistical methods. ROCKA clusters KPIs based on similarities in their underlying shapes despite noises, anomalies, amplitude differences, and phase shifts, and can be helpful in KPI analysis and large-scale anomaly detection. SPF checks if some randomly selected symbolic patterns exist to partition the data instances in the time series. This partition process is executed multiple times, and the partitions are combined to generate the final partition. However, it is challenging to choose a suitable algorithm for traditional statistical methods, and the parameters of those methods are usually tough to tune.

KPI clustering based on deep learning. It is proved by recent researches that deep learning methods can capture features of KPI better [1]. DCN [35] uses the autoencoder structure for dimensionality reduction and uses k-means on the encoded $\mathbf{z}$ for clustering while training the whole model end-to-end. However, most existing deep learning-based time series clustering methods, including DCN, cannot support long-period KPI series well. They are mostly designed for static data, and to the best of our knowledge, no method based on deep learning has been proposed to address the anomaly detection problem.

2) KPI Anomaly Detection: With large-scale Web services becoming more and more popular in recent years, Web anomaly detection methods have received increasing attention. Web anomaly detection is targeted to detect anomalies and protect Web services from attacks [36]–[38] or system failures proactively. Here, we mainly discuss the KPI anomaly detection for Web services.

a) KPI anomaly detection based on traditional methods: Algorithms like [39]–[47] are based on traditional statistical methods. They make some simple assumptions for the KPIs; therefore, efforts must be taken to choose an appropriate algorithm and tune the hyperparameters for each KPI and service. Simple equations used in these models can often not capture the properties of the KPIs in practice. For example, [40] detects the anomalies in search response time KPI with WoW (week-over-week) method. However, there are not only weekly periodicity but also daily periodicity, holidays, and other factors. In such cases, it seems necessary to combine many different detectors. However, simple ensemble of these statistical detectors, like majority vote [45] or normalization [47], does not help a lot [48].

b) KPI anomaly detection based on supervised or semi-supervised learning: Supervised KPI anomaly detection methods can automatically combine detectors according to the input. EDGAS [49] and Opprentice [48] are two examples. EDGAS uses AdaBoost [50] to select the most relevant anomaly data points, while Opprentice ensembles 14 widely-used traditional statistical algorithms to extract features for data points. However, they heavily rely on accurate labels. Getting enough careful labels requires too much manual effort and time, making such supervised methods expensive and impractical. The coverage of anomaly labels is usually far from the requirements of traditional supervised methods. Also, because they may have to execute some time-consuming traditional statistical algorithms, they may take too much time. ATAD [3] combines transfer learning and active learning techniques to transfer knowledge from the source dataset to the target dataset and determine informative labels of a small part of samples from unlabelled datasets. However, using the concept of active learning, its effectiveness heavily relies on the accuracy of the occasionally labeling work. In addition, the algorithms it uses to extract features from KPIs also need to be carefully picked.

c) KPI anomaly detection based on deep learning: For multivariate KPIs, USAD [14], OmniAnomaly [15], and CTF [16] are all advanced unsupervised methods. However, because the nature of anomalies in MTS differs from that of UTS, algorithms proposed by these methods cannot be applied to UTS directly. Donut [1] and Bagel [2] are two unsupervised anomaly detection algorithm based on VAE and CVAE. Bagel dramatically improves the robustness of Donut by using CVAE to incorporate time information and a dropout layer to avoid over-fitting. Buzz [5] makes use of an improved WGAN [51], [52] to greatly improve the robustness of Donut on KPIs with non-Gaussian noises. Nevertheless, these methods are not practical for large-scale Web services for the following two reasons. First, current deep learning-based anomaly detection algorithms suffer from a long initialization time (up to weeks). With the introduction of advanced technologies, *e.g.*, microservice, cloud-native computing, today's Web services frequently deploy software and hardware changes to fix bugs, deploy new features, or improve performance [4], [22]–[24]. Additionally, container instances are frequently created and destroyed to ensure scalability and fault tolerance. After a software/hardware update or a container instance creation, the KPIs of its related entity tend to experience pattern change. Therefore, operators must retrain the anomaly detection model to guarantee its performance. The long initialization time of those deep learning-based KPI anomaly

detection methods makes them not functional for a long time. Secondly, there are hundreds of thousands to millions of physical machines, (micro)service instances, and containers in today's large-scale Web services [18], [53], [54]. Even if operators configure only a few KPIs for each monitored entity, they should conduct anomaly detection for millions of KPIs. Current deep learning-based KPI anomaly detection methods usually train *a dedicated* neural network model for *each KPI*. Moreover, the training cost of *one* neural network model, which digests a large amount of data for model training, is usually high. Consequently, the training overhead of millions of neural network models is prohibitive.

ADT-SHL [55] applies the idea of transfer learning to anomaly detection tasks. It first clusters historical KPIs by similarity, and then all KPIs in each cluster are input into a shared-hidden-layers VAE model. However, it is based on [1], which can not handle KPIs with periodic spikes or with long missing segments well.

III. AnoTransferApproach

A. Design Overview

This paper proposes a novel deep unsupervised KPI anomaly detection algorithm with transfer learning, *AnoTransfer*, to reduce the required initialization time and the training cost. Figure 4 shows the overall framework of *AnoTransfer*. *AnoTransfer* consists of three stages: offline training, transfer learning, and online detection. Historical KPI streams will go through clustering and base model training processes in the offline training stage. In the clustering process, we extract baselines, split them into segments, adjust phase shifts, and utilize VAE with hierarchical agglomerative clustering (HAC) [16], [56] to cluster KPI segments with similar shapes together. The centroid of each cluster constitutes a *shape library*. Then, *AnoTransfer* uses those segments closest to the centroid in each cluster to train a CVAE model as the base model of that shape.

When a new or changed KPI stream arrives in the online environment after service updates, it will first undergo the transfer learning stage before being input into the online detection stage. We match a short period of historical observations of the new-coming KPI to a shape in the shape library and fine-tune its corresponding base model. An automatic transfer strategy selection method is implemented to help the fine-tuning process.

Finally, we use SPOT [21] with hyperparameter sharing to dynamically calculate the anomaly score threshold and use the fine-tuned model to detect anomalies in the online KPI stream.

B. Preprocessing

Commonly, large-scale modern Web services have a massive quantity of monitoring data and monitoring agents. Therefore, it is hard to guarantee that all monitoring data is ideally collected, and data missing is inevitable. According to our observation, the percentage of missing values in a KPI is usually small (less than 10%). *AnoTransfer* uses linear interpolation to fill them based on their adjacent data points. The imputed KPI becomes a time series with a fixed monitoring interval, enabling us to calculate the point-wise distance between two KPIs. Another widely used preprocessing step for time series is standardization. Standardization helps make meaningful comparisons between time series because data collected from different services and systems vary in amplitudes. The standardization process is given by $\mathbf{x}' = \frac{\mathbf{x}-\mu}{\sigma}$, where $\mathbf{x}$ is the raw data, μ and σ are the mean and standard deviation of $\mathbf{x}$, respectively.

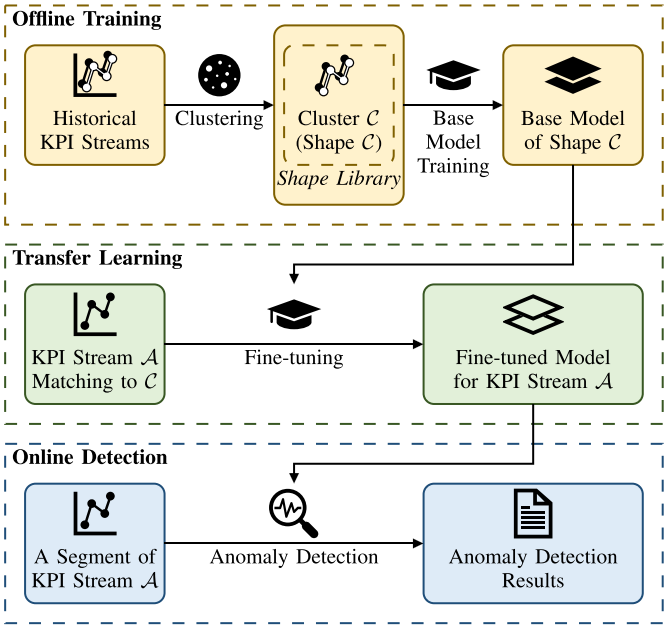

Fig. 4. The overall framework of *AnoTransfer*.

C. Offline Training

1) KPI Clustering: In *AnoTransfer*, the KPI clustering process consists of four steps: baseline extraction, segmentation, phase shift compensation, and clustering.

KPIs with similar shapes can be grouped better when performed on normal patterns. However, noise and anomalies can significantly affect the shape of KPIs and cause miscalculation of shape-based similarity, so we extract smooth baselines from KPIs to represent their underlying shapes. The extreme values are removed first. Extreme values usually have a better chance of being anomalies and the ratio of anomaly points in a KPI is generally less than 5% [19]. Therefore, *AnoTransfer* removes the top 5% data that deviates from the mean value and then uses linear interpolation to fill their vacancies. In this way, extreme anomalies (often huge spikes or dips) are removed and replaced with neighboring normal observations. Then, we use a simple yet effective method to deal with noise. *AnoTransfer* applies the moving average algorithm with a small sliding window on KPIs. The baseline $\mathbf{x}'$ of a KPI $\mathbf{x}$ can be computed as:

$$\begin{aligned} \mathbf{x} &= (x_1, x_2, \ldots, x_n) \\ x'_t &= \frac{1}{W} \sum_{i=t-W+1}^{t} x_i \\ \mathbf{x}' &= \left(x'_W, x'_{W+1}, \ldots, x'_n\right) \end{aligned} \tag{1}$$

where W denotes the window length. For each point x_t in the input, the corresponding point x'_t on the baseline is

the mean of series $(x_{t-W+1}, \ldots, x_t)$. Finally, *AnoTransfer* downsamples the extracted baseline to reduce its sampling rate. Reasonably downsampled KPI can boost the efficiency of clustering without harming the effectiveness of clustering. Please note that *AnoTransfer* only performs baseline extraction (extreme value removal, moving average, *etc*.) in the clustering process. Data used for anomaly detection will not go through this step. Thus, these preprocessing steps will not induce false negatives to anomaly detection results.

After baseline extraction, we extract KPI segments from each baseline. *AnoTransfer* reads the time information of each KPI, discards incomplete days at both ends, and then splits the entire baseline by day. Next, we address the phase shift problem. The purpose of phase shift compensation is to group similar shapes with time lags together, thus making the number of clusters as small as possible. First, we get the pivot $\mathbf{pvt}$ of all the one-day-long segments $\mathcal{S}$ obtained in the previous step by:

$$\mathcal{L}(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_2 \tag{2}$$

$$\mathbf{pvt} = \arg\min_{\mathbf{a}\in\mathcal{S}} \sum_{\mathbf{b}\in\mathcal{S}} \mathcal{L}(\mathbf{a}, \mathbf{b}) \tag{3}$$

We use Euclidean distance $\mathcal{L}$ here, which is simple yet effective in time series classification or clustering [57]. Then, we use normalized cross-correlation (NCC) [20] to calculate the similarity between $\mathbf{pvt}$ and all other curves for all possible phase shifts $s \in [0, n-1]$. To retain as much information as possible, we wrap the KPI around (*i.e.*, move its tail before its head). For $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathbf{y} = (y_1, \ldots, y_n)$, the optimal estimation of their phase shift is computed as:

$$CC_s(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{s} x_t \cdot y_{t+n-s} + \sum_{t=1}^{n-s} x_{s+t} \cdot y_t$$

$$NCC_s(\mathbf{x}, \mathbf{y}) = \frac{CC_s(\mathbf{x}, \mathbf{y})}{\|\mathbf{x}\|_2 \cdot \|\mathbf{y}\|_2}$$

$$s^* = \arg\max_{s\in[0,n-1]} NCC_s(\mathbf{x}, \mathbf{y}) \tag{4}$$

The inner product of the two KPIs is at its maximum when s is closest to the actual phase shift. The range of $NCC(\mathbf{x}, \mathbf{y})$ is $[-1, 1]$. The bigger the NCC value, the more similar two KPIs are. *AnoTransfer* takes the s that maximizes $NCC(\mathbf{x}, \mathbf{y})$ as the phase shift between $\mathbf{x}$ and $\mathbf{y}$.

In the clustering step, we use all the one-day-long KPI segments after phase shift compensation to train a VAE model to learn the characteristics of all shapes as much as possible. *AnoTransfer* then encodes each segment into latent representations using the trained VAE. The process is shown in Figure 5. VAE is used as the feature extractor because it works similarly to the CVAE used in the following training process of the anomaly detection model. Suppose the base KPI and the target KPI are considered similar in shape by the VAE in the clustering process. In that case, the CVAE used in anomaly detection will also encode them into similar latent distributions. *AnoTransfer* inputs all latent representations into the HAC algorithm and gets the clustering result for all KPI segments using Euclidean distance. HAC iteratively merges the closer pair of clusters and moves up the hierarchy until all KPI segments are merged into one cluster. The clustering result can be determined according to the number of required clusters. In *AnoTransfer*, HAC with Ward's linkage is adopted because of the following reasons: 1) Compared to other clustering algorithms, *e.g.*, DBSCAN [19], k-means [35], and k-medoids [58], HAC needs no initial parameters (*e.g.*, the number of clusters or distance thresholds). 2) It is not sensitive to the distance measurement algorithms because it clusters on the rank of distances rather than the value. 3) HAC with Ward's linkage lets each data in the cluster have an equal effect on the inter-cluster distance, making the distance measurement transitive [59]. All the centroids of the clustering result constitute a shape library, for each of them can represent a different type of shape.

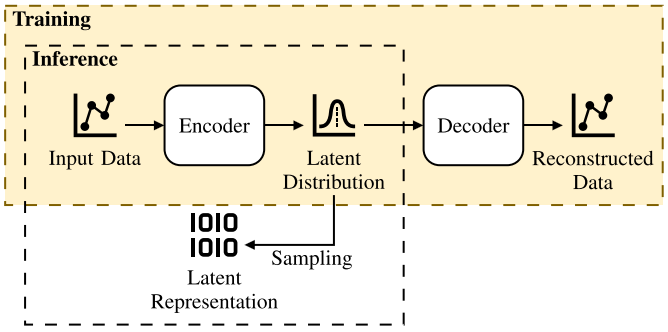

Fig. 5. The VAE model used in KPI clustering. In the training process, the model outputs reconstructed data to optimize itself. In the inference process, the model outputs latent representations used in HAC clustering.

2) Base Model Training: AnoTransfer selects the KPI segments closest to each cluster's centroid to train a CVAE-based anomaly detection model jointly as the base model for each type of shape. We use the sliding window of KPIs as the input of CVAE following [2]. Formally speaking, for a KPI $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, its i-th window is $\mathbf{x}^{(i)} = (x_i, x_{i+1}, \ldots, x_{i+W-1})$, where W denotes the window length. After obtaining the sliding windows for a KPI, *AnoTransfer* encodes the time information of the last point of each window into one-hot vectors. The rule of encoding is illustrated in Figure 6. First, we extract the time information of the last point of the window: minute-of-the-hour, hour-of-the-day, and day-of-the-week. Then, we use one-hot encoding to convert the time information into bit vectors. Motivated by [1], we use missing data injection to augment the training data (*i.e.*, deliberately create some missing points) to increase the difficulty for the model to learn the normal patterns, thereby enhancing the algorithm's robustness.

Figure 7 depicts the architecture of CVAE. At the beginning of training, the time vector $\mathbf{y}$ of a window is randomly dropped out by a dropout layer to prevent over-fitting [60]. Then, the sliding window $\mathbf{x} \in \mathbb{R}^W$ is directly concatenated with the time vector $\mathbf{y} \in \mathbb{R}^Y$ and input into the encoder. The encoder converts it into the mean $\mu_{\mathbf{z}} \in \mathbb{R}^H$ and the standard deviation $\sigma_{\mathbf{z}} \in \mathbb{R}^H$ of the latent distribution. We assume that the prior of the latent distribution is the standard multivariate normal distribution. Next, a latent variable $\mathbf{z} \sim N(\mu_{\mathbf{z}}, \sigma_{\mathbf{z}})$ is sampled from the latent distribution and concatenated with the time vector before being input into the decoder. The decoder outputs

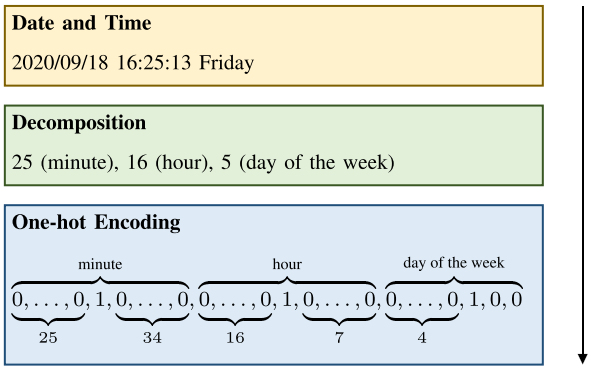

Fig. 6. An example of time information encoding.

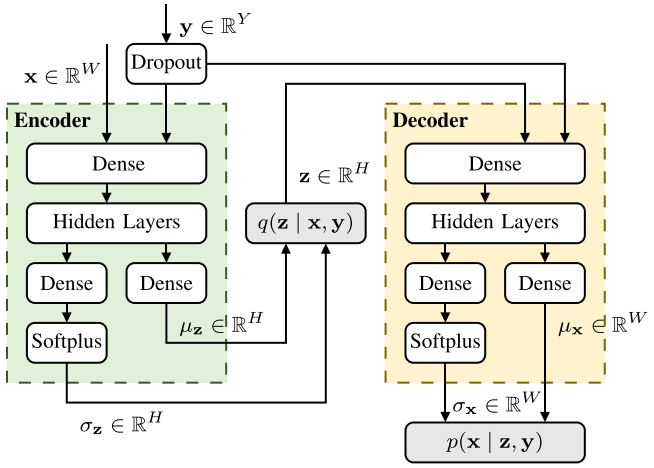

Fig. 7. The neural network architecture of CAVE.

the mean $\mu_\mathbf{x} \in \mathbb{R}^W$ and the standard deviation $\sigma_\mathbf{x} \in \mathbb{R}^W$ of the reconstruction distribution, which completes the forward propagation.

To mitigate the negative impact of missing point in the input as much as possible, *AnoTransfer* adopts the negative modified ELBO (M-ELBO) [2] as the loss function in the backward propagation:

$$\begin{aligned}\tilde{\mathcal{L}}(\mathbf{x}, \mathbf{y}) &= -\text{M-ELBO} \\ &= -\mathbb{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{y})}\left[\sum_{t=1}^{W} \alpha_t \cdot \log p\left(x_t \mid \mathbf{z}, \mathbf{y}\right)\right. \\ &\quad \left. + \beta \cdot \log p\left(\mathbf{z} \mid \mathbf{y}\right) - \log q\left(\mathbf{z} \mid \mathbf{x}, \mathbf{y}\right)\right] \end{aligned} \tag{5}$$

where W is the window length, α is a bit vector, $\alpha_t = 1$ means that x_t is a missing point, and β is the proportion of existing points in the window. The CVAE model is being continuously optimized through the stochastic gradient variational Bayes (SGVB) algorithm [27] and the M-ELBO loss function.

D. Transfer Learning

After obtaining the base model of each cluster, *AnoTransfer* uses the base model to aid the training process in the online environment. For a new-coming or changed KPI (*a.k.a.* target KPI) that needs anomaly detection, *AnoTransfer* takes one-day-long historical observations $\mathbf{x}$ of the KPI and uses Equation (4) to compensate its phase shift with respect to the $\mathbf{pvt}$ picked by Equation (3). The adjusted KPI segment is then input into the clustering VAE to extract its latent representation $\mathbf{z}$, which will be compared against the centroid of each cluster using Euclidean distance (Equation (2)). We select the corresponding model of the cluster that has the smallest distance between $\mathbf{z}$ and its centroid as the base model in transfer learning. *AnoTransfer* utilizes the one-day-long training data from the target KPI to fine-tune the selected base model.

Deep learning models have layered architectures allowing us to utilize pre-trained parameters in the model. Since CVAE is a deep generative model and *AnoTransfer* shares the same prior distribution in both the source domain and the target domain [27], we adopt parameter-based transfer [31] in transfer learning. Parameter-based transfer refers to directly sharing some well-trained parameters between the base and target models. Then we selectively retrain (fine-tune) some of the transferred parameters of the target model to improve its performance. Specifically, the parameters that can be transferred are those in the neural network layers of the encoder and the decoder in CVAE. The outer layers (layers close to the input of the encoder and the output of the decoder) of deep generative models have been seen to capture generic features, while the inner ones (layers close to the output of the encoder and the input of the decoder) focus more on the specific task at hand [61]. Therefore, it is preferred to share the outer layers' parameters and re-initializing the inner layers' parameters. However, when the target KPI $\mathbf{x}$ is very similar to the centroid $\mathbf{c}$ of the cluster it is assigned to, they share not only generic features but also specific ones. Sharing all model parameters can improve the performance of the target model even more. Empirical results in § IV confirm this theoretical implication.

We propose a method to determine the proper transfer method automatically: 1) For target KPIs with a large similarity to the centroid after phase shift compensation, *AnoTransfer* uses the full parameter transfer strategy (*i.e.*, transfer all the parameters in CVAE). 2) For target KPIs with a relatively small similarity to the centroid, *AnoTransfer* uses the partial parameter transfer strategy (*i.e.*, transfer the outer layers' parameters in CVAE). This method also utilizes NCC to calculate the similarity values, which is given by Equation (4) with $s = 0$. A threshold α needs to be chosen by the expertise of human operators. If $NCC(\mathbf{x}, \mathbf{c}) \geq \alpha$, it means that $\mathbf{x}$ and $\mathbf{c}$ are very similar and the full parameter transfer strategy is selected, otherwise the algorithm will choose the partial parameter transfer strategy.

Fine-tuning is performed on the whole transferred model with the full parameter transfer strategy. However, if the partial parameter transfer strategy is selected, the process becomes more complicated. First, a new CVAE model is initialized with random parameters. Then, the outer layers' parameters of the base model are loaded into the target model. To avoid over-fitting, we first freeze the transferred parameters and use the training data from the target KPI to update the newly initialized parameters only. Finally, we unfreeze the transferred parameters and fine-tune the whole model as usual.

E. Online Detection

We use the fine-tuned model to detect anomalies in online KPI streams in the online detection stage. *AnoTransfer* uses

the MCMC missing imputation algorithm [1] to mitigate the impact of missing data as much as possible. The core idea of this algorithm is to let the fine-tuned model reconstruct the missing parts of the input window and then replace the missing data with the reconstructed one. This process will be repeated several times. The intermediate data will keep getting closer to normal values during the whole procedure. Given a sufficiently large number of iterations, we can get a good reconstruction for missing points.

To get anomaly scores, *AnoTransfer* compares the KPI reconstructed by CVAE with the original KPI by calculating the negative reconstruction probability:

$$P\left(\gamma_t = 1 \mid \mathbf{x}, \mathbf{y}\right) = -\mathbb{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{y})}\left[\log p\left(\mathbf{x} \mid \mathbf{z}, \mathbf{y}\right)\right] \tag{6}$$

The higher the anomaly score, the greater the probability that this data point is an anomaly. It is also essential to choose a proper anomaly score threshold dynamically for each KPI in the online environment. Here we use SPOT [21] algorithm with transfer learning to calculate this threshold on KPI data streams. SPOT is an approach to automatically setting anomaly thresholds for streaming UTS based on Extreme Value Theory. It makes no assumption on the data distribution: the main hyperparameters are only the *level* and the *risk*, controlling the number of false positives, which makes it useful in a wide number of situations. We can also use the idea of transfer learning by sharing hyperparameters of SPOT inside each cluster. *AnoTransfer* uses grid search to try some possible combinations of *level* and *risk* to get the optimal performance on the centroid of a cluster. Then, this set of hyperparameters is shared across the same cluster.

IV. Evaluation

In this section, we first introduce the experiment setup, including datasets, performance metrics, and the hyperparameters of *AnoTransfer*. Then, we conduct extensive experiments to evaluate the performance of *AnoTransfer* for addressing the following research questions:

RQ1. How does the performance (effectiveness and efficiency) of *AnoTransfer* compare to the baseline methods?

RQ2. How much initialization time and training time can *AnoTransfer* reduce compared to the non-transfer learning methods?

RQ3. The critical technique contributions of *AnoTransfer* are clustering in the offline training phase, transfer learning adopting, and automatic selection for transfer strategy. How prominently do the three components impact the performance of *AnoTransfer*?

RQ4. The KPI clustering process in *AnoTransfer* consists of four steps: baseline extraction, segmentation, phase shift compensation, and clustering. How well does it perform compared to other clustering methods?

RQ5. How well does the threshold selection method, SPOT, of *AnoTransfer* perform compared to other methods?

RQ6. How does the transfer strategy selection threshold of *AnoTransfer* influence the performance?

RQ7. Case study: Does automatically switching the transfer strategy make sense?

TABLE I
THE DETAILED INFORMATION OF THE DATASETS

	WSD	NAB
# KPIs	210	10
Time Interval	1 min	5 min
# Average Data Points	36471	15863
% Missing Points	0.86%	0
% Anomaly Points	1.58%	9.87%

RQ8. Is *AnoTransfer* robust enough against different types of anomalies?

A. Experiment Setup

1) Datasets and Environment: We use two KPI datasets, the Web service dataset (WSD) and the Numenta Anomaly Benchmark (NAB) dataset [62], to evaluate *AnoTransfer*. WSD contains real-world KPIs collected from three top-tier Internet companies, Baidu, Sogou, and eBay, providing large-scale Web services. Experienced operation engineers have carefully labeled all KPIs in WSD. NAB is a novel and publicly available benchmark for evaluating anomaly detection algorithms in streaming and real-time applications. It contains datasets collected from various companies including AWS, Twitter, *etc*. We choose the dataset from Twitter, which is relatively large in the temporal dimension from NAB, to evaluate *AnoTransfer*. Table I shows more details about the datasets. We use 30% of each dataset as the offline training set and the rest 70% as the online set. For each KPI in the online set, the first one-day-long segment is used to match a shape from the shape library and fine-tune the base model. The rest of the KPI is used as the testing set of anomaly detection. *AnoTransfer* is implemented in Python 3.9 using PyTorch. The source code of *AnoTransfer* is available at.[2] The WSD dataset is available at.[3] All the experiments are run on a server with two 16C32T Intel(R) Xeon(R) Gold 5218 CPU @ 2.30 GHz, one NVIDIA(R) Tesla(R) V100S, and 192 GB RAM.

2) Performance Metrics: The anomaly detection algorithms evaluated in this paper will output an anomaly score for each data point of the testing KPIs and identify the anomalous data points according to the anomaly score threshold. Thus, the anomaly detection problem can be approximated as a classification problem for each data point. We use $F_1 - score$ (F_1 for short) as the performance metric, whose calculation requires True Positives (TP), False Positives (FP), and False Negatives (FN), which is given by:

$$\begin{aligned} \text{precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}} \\ \text{recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \\ F_1 &= 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \end{aligned} \tag{7}$$

We apply three tweaks to classical point-based F_1 according to our observations in the online environment. First, different

[2]https://github.com/anotransfer/AnoTransfer-code/
[3]https://github.com/anotransfer/AnoTransfer-data/

anomaly score thresholds may seriously affect the performance of a detection algorithm. We apply a threshold that can lead to the best F_1 for each algorithm when comparing them [1]: We calculate all (precision, recall) pairs of the algorithm using all possible thresholds and then calculate all F_1 to choose the best one. Second, the anomaly is usually generated by gradual accumulation and will last for a while, so the anomaly start time detected by an algorithm rarely exactly matches the labeled time. Therefore, it is acceptable for the algorithms to trigger an alert for any point in a contiguous anomaly window, if the delay is not too long [1], [2], [14], [15]. As for data points outside of anomaly segments, we treat them point-wisely as usual. Third, missing data points are not counted as anomaly points because missing points are straightforward to recognize by operators and do not need to be detected by anomaly detectors [1]. Therefore, we completely ignore each method's output at all missing points in the evaluation.

3) Hyperparameters: In the offline training stage, the window size of the moving average algorithm is five. On our datasets, a time interval of 10 minutes can yield satisfactory results while keeping the clustering process efficient, so we downsample the input to 10 min/point. For the clustering VAE, both the encoder and the decoder have three dense layers with ReLU activation functions [63]. The number of units of the first two hidden layers is 90, the dimension of the latent variable is 20, and the learning rate is set to 0.0005. For CVAE, the input sliding window size is 120, and the dropout rate of time information is 0.1. The encoder and the decoder have three dense layers with ReLU activation functions. The number of units of the first two hidden layers is 100, the dimension of the latent variable is 8, and the learning rate is 0.001. Each base model is trained by the top 50 closest one-day-long KPI segments to the centroid. We use the Adam optimizer [64] to train both models. In the transfer learning stage, the threshold α for selecting the transfer strategy is 0.7 (see IV for more details). For the hyperparameters of the baseline methods, we use their default values in the experiments.

B. AnoTransfer vs. Baseline Methods (RQ1)

To demonstrate the effectiveness and efficiency of *AnoTransfer*, we compare it with three baseline KPI anomaly detection methods: Bagel [2], ADT-SHL [55], and ATAD [3] (see § II-E.2 for more details). Bagel is a state-of-the-art deep learning-based KPI anomaly detection method. It has outstanding performance when the initialization time is long enough. ADT-SHL and ATAD are two methods that utilize the concept of transfer learning. ADT-SHL is based on the autoencoder structure, while ATAD is based on semi-supervised active learning techniques. Both of them aim to solve the challenge of detecting anomalies in new KPIs with a short initialization time and without anomaly labels. Table II, Figure 8 and Figure 9 shows the F_1 and average time cost of all algorithms on each dataset.

AnoTransfer outperforms all baselines in effectiveness and efficiency on both datasets. *AnoTransfer*'s F_1 are 0.915 and 0.972 on WSD and NAB, respectively, while the best F_1 of baselines are 0.802 and 0.966, which is lower than *AnoTransfer*. As for efficiency, we evaluate each algorithm's training time for new/changed KPIs. *AnoTransfer* is significantly faster than baseline methods with 0.533 sec/KPI and 0.398 sec/KPI on each dataset, respectively. It takes Bagel and ADT-SHL around 10 seconds to train/fine-tune the anomaly detection model for a target KPI until the model converges. ATAD needs even more training time to successfully adapt to target KPIs, rendering it impractical in our scenario.

We will try to analyze the reasons for this result next. On WSD, *AnoTransfer* outperforms others by 14.09% to 22.16%. Bagel needs a long initialization time to get good performance. However, the training data acting as target KPIs (one-day-long) in this experiment are not long enough for Bagel, so its average best F_1 is not as good as *AnoTransfer*, and takes a longer time than *AnoTransfer* to converge. ADT-SHL shows slightly better performance because of its transfer learning techniques. However, it is based on [1], which can not handle KPIs with periodic spikes or with long-missing segments well, resulting in a suboptimal performance than *AnoTransfer*. Additionally, with multiple

TABLE II
PERFORMANCE OF ANOTRANSFER AND OTHER BASELINE ALGORITHMS

	WSD		NAB	
	F_1	Time (s)	F_1	Time (s)
Bagel	0.749	9.071	0.966	8.498
ADT-SHL	0.802	11.988	0.769	11.522
ATAD	0.773	53.900	0.775	47.286
AnoTransfer	**0.915**	**0.533**	**0.972**	**0.398**

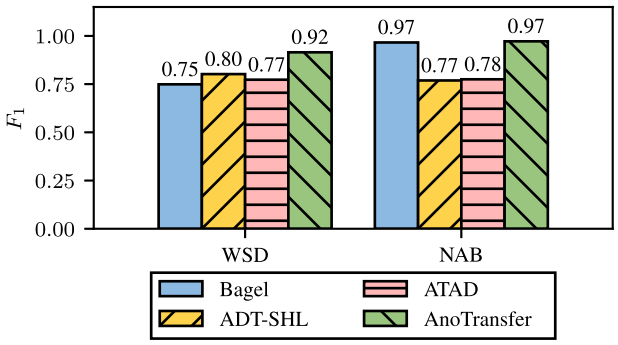

Fig. 8. Overall performance of KPI anomaly detection algorithms.

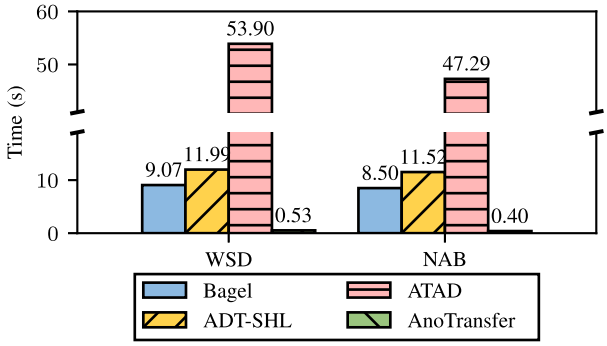

Fig. 9. Average training time for a single new/changed KPI of each algorithm.

convolutional networks, its Shared-Hidden-Layers VAE has a more complex structure than *AnoTransfer*, increasing its training time overhead. ATAD's performance is not satisfactory, either. The concept of active learning makes its effectiveness heavily rely on the accuracy of the occasionally labeling work. In addition, the algorithms it uses to extract features from KPIs also need to be carefully picked. Those features we selected for one dataset may not be suitable for others. Too many complex feature extraction algorithms also make ATAD slower than *AnoTransfer*. On NAB, ADT-SHL's performance is not as good as the WSD because KPIs in this dataset are very noisy and have many periodic local patterns such as spikes and dives, while *AnoTransfer* can handle this type of KPIs well thanks to the CVAE. *AnoTransfer*'s performance is roughly the same as Bagel's, but *AnoTransfer* costs much less time to train.

C. Effect of Transfer Learning (RQ2)

In this section, we conduct two experiments to determine how much of the required training time and initialization time can be reduced using transfer learning. Here we choose Bagel as the comparison non-transfer method for it has the best efficiency of baselines. Bagel trains a new model for each target KPI while *AnoTransfer* fine-tunes the corresponding base model for each target KPI.

In the first experiment, we gradually increase the length of training data (initialization time) and monitor *AnoTransfer* and Bagel's performance on the WSD dataset. We take enough training time to let both models converge. Figure 10 shows that Bagel's performance improves rapidly before 1000 minutes of training data, and its F_1 increases steadily after that. Bagel cannot reach its maximum potential even when trained with 4200 minutes of the target KPI. On the other hand, *AnoTransfer*'s performance improves rapidly before 1500 minutes of training data. However, it remains at the same level when the number of KPIs grows bigger, which means we can get a stable performance even using only one-day-long training data.

In the second experiment, we test *AnoTransfer* and Bagel's training time when facing a vast number of target KPIs. We use a much bigger real-world KPI dataset than WSD and NAB in this experiment.[4] This dataset is collected from ByteDance, a top-tier global content service provider providing services for one billion+ monthly active users (MAU), and contains over 40000 KPIs. However, operators cannot provide anomaly labels because of the considerable labeling overhead brought by the large quantity. So, unfortunately, we cannot use it in anomaly detection experiments. Figure 11 shows that the training cost of *AnoTransfer* is significantly lower than existing methods like Bagel. When the number of target KPIs grows large, the training time of Bagel escalates very fast, while the cost of *AnoTransfer* can stay at a relatively low level.

D. Contributions of Components (RQ3)

To show the effects of three key techniques in *AnoTransfer*: 1) clustering; 2) transfer learning; 3) automatic transfer strategy selection, we reconfigure *AnoTransfer* to create four

[4]https://github.com/omni-cluster/OmniCluster-dataset/

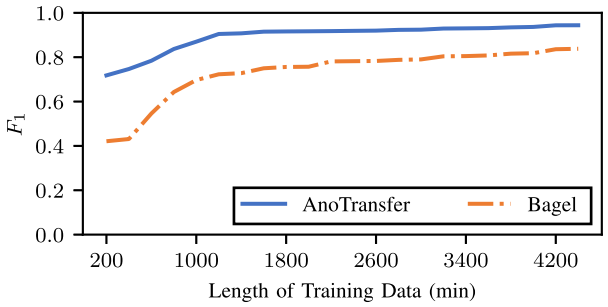

Fig. 10. The performance of *AnoTransfer* and Bagel with different initialization time. *AnoTransfer* can get satisfactory results using shorter training data than Bagel.

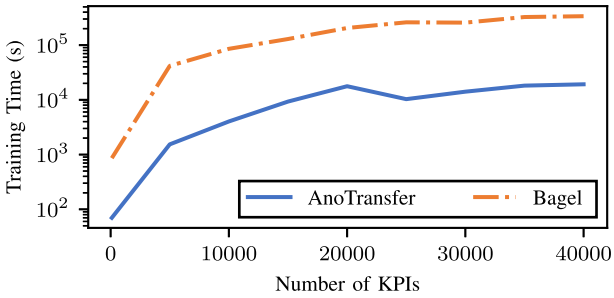

Fig. 11. The training time of *AnoTransfer* and Bagel on different numbers of KPIs.

TABLE III
COMPARISON WITH MODEL VARIATIONS

	WSD	NAB
C1	0.680	**0.972**
C2	0.718	0.926
C3	0.874	0.969
C4	0.783	0.967
AnoTransfer	**0.915**	**0.972**

variants **C1**-**C4**, described as follows: **C1**: *AnoTransfer* w/o clustering: Only one base model is used for transfer learning, and the KPI segments used to train the base model are randomly picked from the whole dataset. **C2**: *AnoTransfer* w/o transfer learning: The base model of each cluster is used to detect anomalies in target KPIs directly, without parameter transfer or fine-tuning. **C3**: *AnoTransfer* using only the full parameter transfer strategy. **C4**: *AnoTransfer* using only the partial parameter transfer strategy. Table III and Figure 12 shows the performance of *AnoTransfer* and its variants.

1) Effect of Clustering: With an F_1 of only 0.680, the performance of "*AnoTransfer* w/o Clustering" is far from satisfactory on WSD. This result indicates clustering can effectively group KPIs with similar shapes together, making it easy to match a target KPI with a suitable base model, improving transfer learning effectiveness. As for the NAB dataset, KPIs in it are all very similar, and they can reasonably be grouped into the same cluster, which leads to the same F_1 as *AnoTransfer*. Generally, it is not easy to guarantee that all the KPIs in real-world use cases are close, making clustering essential in *AnoTransfer*.

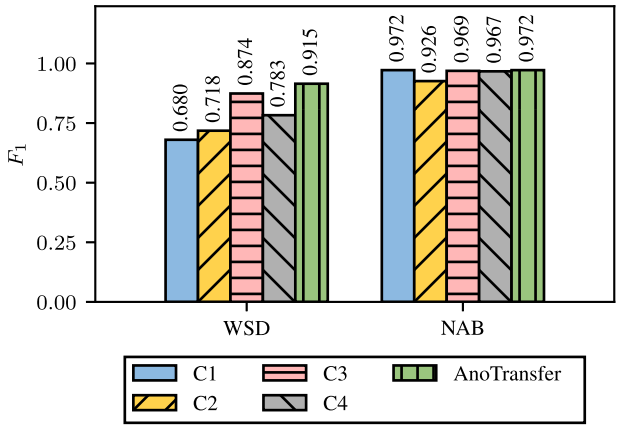

Fig. 12. The performance of *AnoTransfer* and its variants.

2) Effect of Transfer Learning: With "*AnoTransfer* w/o Transfer Learning", WSD is grouped into 10 clusters. Although the target KPI should be reasonably similar to its matching cluster, there are still many local variations, *e.g.*, different noise patterns, minor shape changes. These variations make the F_1 relatively poor if we use the base model directly in anomaly detection tasks without fine-tuning. On the contrary, as mentioned above, KPIs in NAB are very similar, so only using the base model is enough in this case. Thus, "*AnoTransfer* w/o Transfer Learning" on NAB achieves similar performance as *AnoTransfer*. However, it is rare to have a cluster whose all target KPIs are close to its centroid. Therefore, it is indispensable to transfer model parameters and fine-tune the target model.

3) Effect of Automatic Transfer Strategy Selection: When using a fixed transfer strategy instead of automatically switching two strategies, *AnoTransfer* using only the full parameter transfer strategy can perform better than *AnoTransfer* using only the partial parameter transfer strategy. It is because more parameters can carry more valuable knowledge learned from the offline training stage. Most of the target KPIs in our dataset are close to the centroid, so using the full parameter transfer strategy can take advantage of all information in the base model. However, transferring all parameters can harm the performance if the target KPI is relatively far from the centroid. We discover that a small number of target KPIs in our dataset have relatively large shape changes compared to the centroid. This can be caused by the variety of KPIs or errors in the class assignment step. By automatically selecting the best transfer strategy according to the distance between the target KPI and the centroid, *AnoTransfer* gets the highest F_1.

E. Evaluation of KPI Clustering (RQ4)

To prove *AnoTransfer*'s KPI clustering algorithm is better than existing ones in our scenario, we select three UTS clustering algorithms in this section: ROCKA [19], SPF [34], and DCN [35]. The architectures of these approaches are various. ROCKA and SPF are based on traditional statistical methods, while DCN and *AnoTransfer* are based on deep learning. Since NAB can be reasonably grouped into a single cluster, we only use WSD in this experiment. We split our dataset into one-day-long segments without other preprocessing steps before input into each baseline algorithm. The clustering results given by each clustering algorithm are then used in the base model training and transfer learning stage of *AnoTransfer*. We evaluate the clustering performance of the four algorithms above by the final performance in anomaly detection. The experimental results are shown in Figure 13. *AnoTransfer*'s KPI clustering algorithm outperforms others by 14.95% to 31.09%. This is because ROCKA and SPF are not resistant to noise in the dataset. Their clustering results are not suitable for deep learning-based algorithms like *AnoTransfer*, either. DCN theoretically could resist noise by using the autoencoder structure. However, it is hard to fully converge (it has a few billion parameters), and thus its latent representations may be inaccurate. The latent distribution in *AnoTransfer*'s clustering VAE is robust to noise and low-dimensional to converge, thus the most accurate.

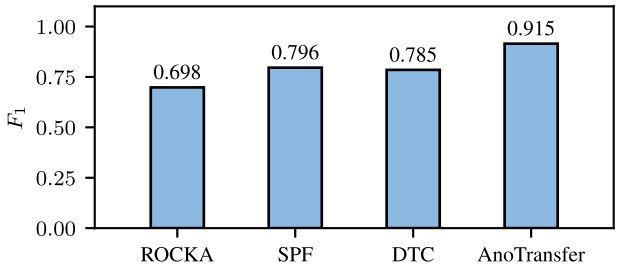

Fig. 13. The performance of *AnoTransfer*'s transfer learning component combined with different clustering algorithms.

F. Evaluation of Anomaly Score Threshold Selection (RQ5)

Although we can use the best threshold (which will yield the best F_1 on each target KPI) to evaluate the theoretical best performance of *AnoTransfer* on each dataset, it is not feasible in the real-world online environment. It is impossible to search for the best threshold for data yet to come exhaustively. We conduct an experiment to evaluate *AnoTransfer*'s performance on online stream data. *AnoTransfer* uses SPOT with shared hyperparameters for each cluster to calculate the anomaly score threshold dynamically. We compare this design with the other two methods: static thresholds and SPOT without hyperparameter sharing. With static thresholds, we determine those data points with the highest 5% anomaly score as anomaly points. When using SPOT without hyperparameter sharing, we set *risk* and *level* to their default values. The experimental result is shown in Table IV and Figure 14. *AnoTransfer* outperforms SPOT with fixed hyperparameters by 0.16 and is not far behind the best F_1. Static thresholds perform the worst, which is not practical in real-world scenarios. This indicates that by sharing a set of hyperparameters across target KPIs in the same cluster, we can successfully determine the anomaly score threshold for online data without degrading F_1 too much.

G. Effect of Transfer Strategy Selection Threshold (RQ6)

To study how the transfer strategy selection threshold α influences *AnoTransfer*'s performance, we conduct an

TABLE IV

PERFORMANCE OF ANOMALY SCORE THRESHOLD SELECTION METHODS

	WSD	NAB
Static	0.402	0.522
SPOT (Fixed)	0.668	0.936
Best	**0.915**	**0.972**
AnoTransfer (SPOT (Shared))	0.828	0.936

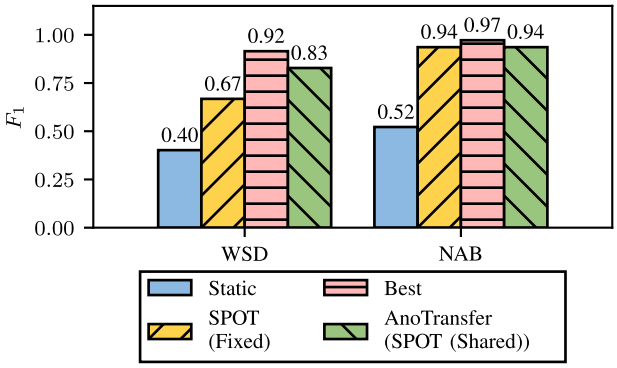

Fig. 14. The performance of *AnoTransfer* using different anomaly score threshold selection methods.

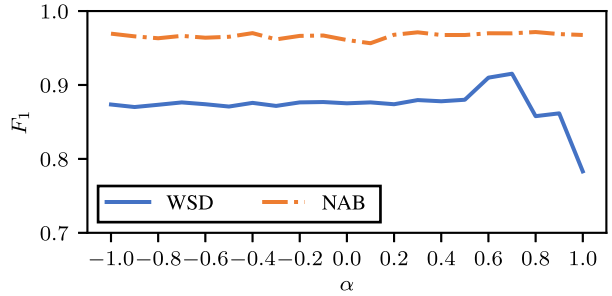

Fig. 15. The performance of different α.

experiment to evaluate *AnoTransfer* with different values of α. Figure 15 shows the experimental result. Recall that the larger the α, the more target models will be fine-tuned with the partial parameter transfer strategy. The performance of *AnoTransfer* peeks at an α of 0.7 on WSD. We can see that when α is greater than 0.7, F_1 has a noticeable drop, for a too-large α will discard too much information in base models. When α is smaller than 0.5, the performance of *AnoTransfer* becomes stable, which means that nearly all KPIs belonging to the same cluster have an NCC value greater than 0.5 to the centroid. Thus the full parameter transfer strategy is always selected. So we set α to 0.7 in *AnoTransfer* empirically. On NAB, the value of α does not affect F_1. This is because KPIs in NAB have relatively simple overall patterns than those in WSD, and *AnoTransfer* can handle noise and periodic spikes well. Using only the partial parameter transfer strategy can already get a satisfactory result. Nevertheless, *AnoTransfer* can achieve good performance with an extensive range of α on both datasets.

H. Case Study: Effect of Transfer Strategy Selection (RQ7)

As described in § III-D, there are two ways to transfer model parameters for CVAE. To prove the effectiveness of automatic transfer strategy selection, we conducted an experiment on three KPIs from WSD, $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$. We select $\mathcal{C}$ as the base KPI and $\mathcal{A}$, $\mathcal{B}$ as the target KPIs. The summary of these KPIs is displayed in Table V. The NCC value between $\mathcal{A}$ and $\mathcal{C}$ is 0.927, which means that $\mathcal{A}$ and $\mathcal{C}$ have a good similarity, whereas $NCC(\mathcal{B},\mathcal{C})$ is 0.254, indicating that the similarity between $\mathcal{B}$ and $\mathcal{C}$ is relatively poor.

TABLE V

SUMMARY OF $\mathcal{A}$, $\mathcal{B}$ AND $\mathcal{C}$

	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$
# Data Points	49842	49842	43932
% Missing Points	1.269%	1.22%	8.858%
% Anomaly Points	3.107%	1.04%	3.787%

TABLE VI

PERFORMANCE ON $\mathcal{A}$ AND $\mathcal{B}$ WITH DIFFERENT STRATEGIES

	$\mathcal{A}$	$\mathcal{B}$
S1	0.781	0.722
S2	0.645	0.828
S3	0.464	0.403
S4	**0.803**	**0.931**

This experiment tests the performance of anomaly detection with four strategies **S1**-**S4**: **S1:** full parameter transfer strategy of *AnoTransfer*; **S2:** partial parameter transfer strategy of *AnoTransfer*; **S3:** no transfer learning (short): use the first one-day-long segment of the target KPI as the training set to train a new model for each KPI; **S4:** no transfer learning (long): use a 4-week-long segment of the target KPI as the training set to train a new model for each KPI (train a new model with enough training data). The results are shown in Table VI and Figure 16. Since $\mathcal{A}$ and $\mathcal{C}$ are very similar, the performance of the full parameter transfer strategy is better than the partial parameter transfer strategy when using transfer learning. Because of the large difference between $\mathcal{B}$ and $\mathcal{C}$, the full parameter transfer strategy will lead to an increase in the false alarm rate. As a result, the final F_1 is not as good as that of the partial parameter transfer strategy. We get the worst performance when no transfer learning is performed, and only a small amount of data is used for training. When using no transfer learning with long enough training data, we get the best performance on both $\mathcal{A}$ and $\mathcal{B}$. The results produced by the best transfer strategy of $\mathcal{A}$ and $\mathcal{B}$ are not very far behind S4. Considering the above results, it is essential for *AnoTransfer* to automatically select the best transfer strategy.

I. The Robustness of AnoTransfer (RQ8)

Recent researches [65], [66] stated that many existing benchmarks for time series anomaly detection algorithms are flawed. According to [66], anomalies in time series can be roughly categorized into five types: global point outliers, contextual point outliers, shapelet outliers, seasonal outliers, and trend outliers. Flawed datasets usually lack one or more types of anomalies, or have too many inaccurate

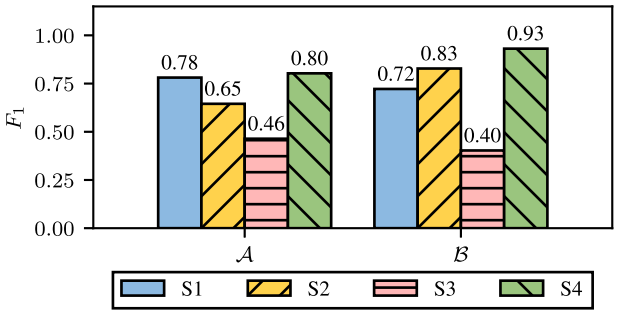

Fig. 16. The performance of *AnoTransfer* on $\mathcal{A}$ and $\mathcal{B}$ with each strategy.

anomaly labels. Because of these flaws, it is believed that many published comparisons of anomaly detection algorithms may be unreliable [65]. To evaluate whether *AnoTransfer* is robust enough against various types of UTS anomalies, we use the strategies proposed by [66] to construct a synthetic dataset including all five anomaly types. Specifically, we first adopt sinusoidal waves of different frequencies and amplitudes as the base shapelet and then inject various types of anomalies. Our synthetic dataset contains 300 UTS, each of which has 30000 data points. The point-wise anomaly rate, *i.e.*, $\frac{\#\ anomaly\ data\ points}{\#\ total\ data\ points}$, is 13.93%. This dataset is also publicly available at the WSD repository (see § IV-A). We use 30% of the dataset as the offline training set and the rest 70% as the online set. *AnoTransfer*'s average best F_1 on this dataset is 0.962. Based on this result, we can determine that *AnoTransfer* is robust and can robustly detect all types of anomalies.

In general, *AnoTransfer* achieves satisfactory anomaly detection performance with high efficiency on two real-world datasets (WSD and NAB) and a synthetic dataset. It is robust and can be applied to various real-world scenarios without worrying about the complexity of data.

V. Conclusion

The deep learning-based KPI anomaly detection methods suffer from high training overhead and long initialization time, making them inappropriate for today's large-scale Web services. To the best of our knowledge, we are among the first to identify the two key drawbacks of these methods. Built upon a state-of-the-art unsupervised neural network structure, CVAE, *AnoTransfer* is an unsupervised KPI anomaly detection algorithm that can effectively shorten initialization time and significantly reduce the training time needed by using the transfer learning technique. We propose a new KPI clustering algorithm based on VAE and HAC to make transfer learning perform better. In addition, SPOT with shared hyperparameters is integrated into the model to calculate the anomaly score threshold on online stream data dynamically, so operators do not have to set thresholds manually. Our experiments using real-world data from large Internet companies show that compared to various baseline methods, *AnoTransfer* can reduce the average initialization time by 65.71%, and improve the training efficiency by 50.62 times, respectively. We believe *AnoTransfer* is especially useful for large Web service providers, for they usually have millions of entities to be monitored and a large number of changed KPIs every day. *AnoTransfer* can make the KPI anomaly detection system adapt to changed KPIs as quickly as possible with affordable cost. The idea of transfer learning-based warmup can also be adopted by many other deep learning-based tasks, *e.g.*, MTS anomaly detection, intrusion detection, log anomaly detection, and trace anomaly detection, to accelerate their training process.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable feedback.

References

[1] H. Xu *et al.*, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proc. World Wide Web Conf. (WWW)*, P. Champin, F. Gandon, M. Lalmas, and P. G. Ipeirotis, Eds., Lyon, France, Apr. 2018, pp. 187–196, doi: 10.1145/3178876.3185996.

[2] Z. Li, W. Chen, and D. Pei, "Robust and unsupervised KPI anomaly detection based on conditional variational autoencoder," in *Proc. IEEE 37th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Orlando, FL, USA, Nov. 2018, pp. 1–9, doi: 10.1109/PCCC.2018.8710885.

[3] X. Zhang *et al.*, "Cross-dataset time series anomaly detection for cloud systems," in *Proc. Annu. Tech. Conf. USENIX ATC*, D. Malkhi and D. Tsafrir, Eds., Renton, WA, USA, 2019, pp. 1063–1076. [Online]. Available: https://www.usenix.org/conference/atc19/presentation/zhang-xu

[4] M. Ma, S. Zhang, D. Pei, X. Huang, and H. Dai, "Robust and rapid adaption for concept drift in software system anomaly detection," in *Proc. IEEE 29th Int. Symp. Softw. Rel. Eng. (ISSRE)*, S. Ghosh, R. Natella, B. Cukic, R. S. Poston, and N. Laranjeiro, Eds., Memphis, TN, USA, Oct. 2018, pp. 13–24, doi: 10.1109/ISSRE.2018.00013.

[5] W. Chen *et al.*, "Unsupervised anomaly detection for intricate KPIs via adversarial training of VAE," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Paris, France, Apr. 2019, pp. 1891–1899, doi: 10.1109/INFOCOM.2019.8737430.

[6] J. Bu *et al.*, "Rapid deployment of anomaly detection models for large number of emerging KPI streams," in *Proc. IEEE 37th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Orlando, FL, USA, Nov. 2018, pp. 1–8, doi: 10.1109/PCCC.2018.8711315.

[7] Z. Shang, Y. Zhang, X. Zhang, Y. Zhao, Z. Cao, and X. Wang, "Time series anomaly detection for KPIs based on correlation analysis and HMM," *Appl. Sci.*, vol. 11, no. 23, p. 11353, Nov. 2021.

[8] L. Dai *et al.*, "SDFVAE: Static and dynamic factorized VAE for anomaly detection of multivariate CDN KPIs," in *Proc. Web Conf.*, J. Leskovec, M. Grobelnik, M. Najork, J. Tang, and L. Zia, Eds., Ljubljana, Slovenia, Apr. 2021, pp. 3076–3086, doi: 10.1145/3442381.3450013.

[9] A. Abdulaal, Z. Liu, and T. Lancewicki, "Practical approach to asynchronous multivariate time series anomaly detection and localization," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining (KDD)*, F. Zhu, B. C. Ooi, and C. Miao, Eds., Aug. 2021, pp. 2485–2494, doi: 10.1145/3447548.3467174.

[10] 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*, F. Zhu, B. C. Ooi, and C. Miao, Eds., Singapore, Aug. 2021, pp. 3220–3230, doi: 10.1145/3447548.3467075.

[11] L. Shen, Z. Yu, Q. Ma, and J. T. Kwok, "Time series anomaly detection with multiresolution ensemble decoding," in *Proc. 35th AAAI Conf. Artif. Intell., 33rd Conf. Innov. Appl. Artif. Intell. (IAAI), 11th Symp. Educ. Adv. Artif. Intell. (EAAI)*, 2021, pp. 9567–9575. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17152

[12] H. Ye, X. Ma, Q. Pan, H. Fang, H. Xiang, and T. Shao, "An adaptive approach for anomaly detector selection and fine-tuning in time series," in *Proc. 1st Int. Workshop Deep Learn. Pract. High-Dimensional Sparse Data*, Aug. 2019, pp. 1–7.

[13] Z. He *et al.*, "A spatiotemporal deep learning approach for unsupervised anomaly detection in cloud systems," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Oct. 16, 2020, doi: 10.1109/TNNLS.2020.3027736.

[14] 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*, R. Gupta, Y. Liu, J. Tang, and B. A. Prakash, Eds., Virtual Event, CA, USA, Aug. 2020, pp. 3395–3404, doi: 10.1145/3394486.3403392.

[15] 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*, A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, and G. Karypis, Eds., Anchorage, AK, USA, Jul. 2019, pp. 2828–2837, doi: 10.1145/3292500.3330672.

[16] M. Sun *et al.*, "CTF: Anomaly detection in high-dimensional time series with coarse-to-fine model transfer," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Vancouver, BC, Canada, May 2021, pp. 1–10, doi: 10.1109/INFOCOM42981.2021.9488755.

[17] M. Ma *et al.*, "Jump-starting multivariate time series anomaly detection for online service systems," in *Proc. Annu. Tech. Conf. USENIX ATC*, I. Calciu and G. Kuenning, Eds., 2021, pp. 413–426. [Online]. Available: https://www.usenix.org/conference/atc21/presentation/ma

[18] Y. Su *et al.*, "Detecting outlier machine instances through Gaussian mixture variational autoencoder with one dimensional CNN," *IEEE Trans. Comput.*, vol. 71, no. 4, pp. 892–905, Apr. 2022.

[19] Z. Li, Y. Zhao, R. Liu, and D. Pei, "Robust and rapid clustering of KPIs for large-scale anomaly detection," in *Proc. IEEE/ACM 26th Int. Symp. Quality Service (IWQoS)*, Banff, AB, Canada, Jun. 2018, pp. 1–10, doi: 10.1109/IWQoS.2018.8624168.

[20] J. Paparrizos and L. Gravano, "K-shape: Efficient and accurate clustering of time series," *ACM SIGMOD Rec.*, vol. 45, no. 1, pp. 69–76, Jun. 2016, doi: 10.1145/2949741.2949758.

[21] 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*, Halifax, NS, Canada, Aug. 2017, pp. 1067–1075, doi: 10.1145/3097983.3098144.

[22] Y. Zhang *et al.*, "Understanding and detecting software upgrade failures in distributed systems," in *Proc. ACM SIGOPS 28th Symp. Oper. Syst. Princ.*, R. van Renesse and N. Zeldovich, Eds., Koblenz, Germany, Oct. 2021, pp. 116–131, doi: 10.1145/3477132.3483577.

[23] Z. Li *et al.*, "Gandalf: An intelligent, end-to-end analytics service for safe deployment in large-scale cloud infrastructure," in *Proc. 17th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, R. Bhagwan and G. Porter, Eds., Santa Clara, CA, USA, 2020, pp. 389–402. [Online]. Available: https://www.usenix.org/conference/nsdi20/presentation/li

[24] N. Zhao *et al.*, "Identifying bad software changes via multimodal anomaly detection for online service systems," in *Proc. 29th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng. (ESEC/FSE)*, D. Spinellis, G. Gousios, M. Chechik, and M. D. Penta, Eds., Athens, Greece, Aug. 2021, pp. 527–539, doi: 10.1145/3468264.3468543.

[25] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Found. Trends Mach. Learn.*, vol. 12, no. 4, pp. 307–392, 2019, doi: 10.1561/2200000056.

[26] G. Wu, H. Zhang, Y. He, X. Bao, L. Li, and X. Hu, "Learning Kullback–Leibler divergence-based Gaussian model for multivariate time series classification," *IEEE Access*, vol. 7, pp. 139580–139591, 2019, doi: 10.1109/ACCESS.2019.2943474.

[27] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Montreal, CA, Canada, Dec. 2015, pp. 3483–3491.

[28] E. Fetaya, J. Jacobsen, W. Grathwohl, and R. S. Zemel, "Understanding the limitations of conditional generative models," in *Proc. 8th Int. Conf. Learn. Represent. (ICLR)*, Addis Ababa, Ethiopia, Apr. 2020, pp. 1–6. [Online]. Available: https://openreview.net/forum?id=r1lPleBFvH

[29] S. Bozinovski, "Reminder of the first paper on transfer learning in neural networks, 1976," *Informatica*, vol. 44, no. 3, Sep. 2020, pp. 291–302. [Online]. Available: http://www.informatica.si/index.php/informatica/article/view/2828

[30] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.

[31] F. Zhuang *et al.*, "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021, doi: 10.1109/JPROC.2020.3004555.

[32] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He, "Supervised representation learning: Transfer learning with deep autoencoders," in *Proc. 24th Int. Joint Conf. Artif. Intell. (IJCAI)*, Q. Yang and M. J. Wooldridge, Eds., Buenos Aires, Argentina, 2015, pp. 4119–4125. [Online]. Available: http://ijcai.org/Abstract/15/578

[33] Y. Zhao, R. Rossi, and L. Akoglu, "Automatic unsupervised outlier model selection," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 1–14.

[34] X. Li, J. Lin, and L. Zhao, "Linear time complexity time series clustering with symbolic pattern forest," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, S. Kraus, Ed., Macao, China, Aug. 2019, pp. 2930–2936, doi: 10.24963/ijcai.2019/406.

[35] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards K-means-friendly spaces: Simultaneous deep learning and clustering," in *Proc. 34th Int. Conf. Mach. Learn.*, D. Precup and Y. W. Teh, Eds., vol. 70, Aug. 2017, pp. 3861–3870. [Online]. Available: http://proceedings.mlr.press/v70/yang17b.html

[36] B. A. Tama, L. Nkenyereye, S. M. R. Islam, and K. Kwak, "An enhanced anomaly detection in web traffic using a stack of classifier ensemble," *IEEE Access*, vol. 8, pp. 24120–24134, 2020, doi: 10.1109/ACCESS.2020.2969428.

[37] S. Carta, A. S. Podda, D. R. Recupero, and R. Saia, "A local feature engineering strategy to improve network anomaly detection," *Future Internet*, vol. 12, no. 10, p. 177, Oct. 2020, doi: 1999-5903/12/10/177.

[38] Z. Cheng, B. Cui, and J. Fu, "A novel web anomaly detection approach based on semantic structure," in *Security and Privacy in Social Networks and Big Data* (Communications in Computer and Information Science), vol. 1298, Y. Xiang, Z. Liu, and J. Li, Eds., Tianjin, China, 2020, pp. 20–33, doi: 10.1007/978-981-15-9031-3_2.

[39] S. Zhang *et al.*, "FUNNEL: Assessing software changes in web-based services," *IEEE Trans. Services Comput.*, vol. 11, no. 1, pp. 34–48, Jan. 2018, doi: 10.1109/TSC.2016.2539945.

[40] Y. Chen, R. Mahajan, B. Sridharan, and Z.-L. Zhang, "A provider-side view of web search response time," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, D. M. Chiu, J. Wang, P. Barford, and S. Seshan, Eds., Hong Kong, Aug. 2013, pp. 243–254, doi: 10.1145/2486001.2486035.

[41] F. Knorn and D. J. Leith, "Adaptive Kalman filtering for anomaly detection in software appliances," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM)*, Apr. 2008, pp. 1–6.

[42] H. Yan *et al.*, "Argus: End-to-end service anomaly detection and localization from an ISP's point of view," in *Proc. IEEE INFOCOM*, A. G. Greenberg and K. Sohraby, Eds., Orlando, FL, USA, Mar. 2012, pp. 2756–2760, doi: 10.1109/INFCOM.2012.6195694.

[43] W. Lu and A. A. Ghorbani, "Network anomaly detection based on wavelet analysis," *EURASIP J. Adv. Signal Process.*, vol. 2009, no. 1, pp. 1–16, Dec. 2008, doi: 10.1155/2009/837601.

[44] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognit.*, vol. 58, pp. 121–134, Oct. 2016, doi: 10.1016/j.patcog.2016.03.028.

[45] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "MAWILab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in *Proc. 6th Int. Conf.*, J. C. de Oliveira, M. Ott, T. G. Griffin, and M. Médard, Eds., Philadelphia, PA, USA, Nov. 2010, pp. 1–12, doi: 10.1145/1921168.1921179.

[46] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: Methods, evaluation, and applications," in *Proc. 3rd ACM SIGCOMM Conf. Internet Meas.*, Miami Beach, FL, USA, 2003, pp. 234–247, doi: 10.1145/948205.948236.

[47] S. Shanbhag and T. Wolf, "Accurate anomaly detection through parallelism," *IEEE Netw.*, vol. 23, no. 1, pp. 22–28, Jan. 2009, doi: 10.1109/MNET.2009.4804320.

[48] D. Liu *et al.*, "Opprentice: Towards practical and automatic anomaly detection through machine learning," in *Proc. Internet Meas. Conf.*, K. Cho, K. Fukuda, V. S. Pai, and N. Spring, Eds., Tokyo, Japan, Oct. 2015, pp. 211–224, doi: 10.1145/2815675.2815679.

[49] N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, L. Cao, C. Zhang, T. Joachims, G. I. Webb, D. D. Margineantu, and G. Williams, Eds., Sydney, NSW, Australia, Aug. 2015, pp. 1939–1947, doi: 10.1145/2783258.2788611.

[50] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proc. 2nd Eur. Conf. Comput. Learn. Theory*, in Lecture Notes in Computer Science, vol. 904, P. M. B. Vitányi, Ed., Barcelona, Spain, 1995, pp. 23–37, doi: 10.1007/3-540-59119-2_166.

[51] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. 34th Int. Conf. Mach. Learn.*, in Proceedings of Machine Learning Research, vol. 70, D. Precup and Y. W. Teh, Eds., Sydney, NSW, Australia, Aug. 2017, pp. 214–223. [Online]. Available: http://proceedings.mlr.press/v70/arjovsky17a.html

[52] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, I. Guyon *et al.*, Eds., Long Beach, CA, USA, Jun. 2017, pp. 5767–5777. [Online]. Available: http://papers.nips.cc/paper/7159-improved-training-of-wasserstein-gans
[53] C. Guo *et al.*, "Pingmesh: A large-scale system for data center network latency measurement and analysis," in *Proc. ACM Conf. Special Interest Group Data Commun.*, S. Uhlig, O. Maennel, B. Karp, and J. Padhye, Eds., London, U.K., Aug. 2015, pp. 139–152, doi: 10.1145/2785956.2787496.
[54] A. Singh *et al.*, "Jupiter rising: A decade of clos topologies and centralized control in Google's datacenter network," in *Proc. ACM Conf. Special Interest Group Data Commun.*, S. Uhlig, O. Maennel, B. Karp, and J. Padhye, Eds., London, U.K., Aug. 2015, pp. 183–197, doi: 10.1145/2785956.2787508.
[55] X. Duan, N. Chen, and Y. Xie, "Intelligent detection of large-scale KPI streams anomaly based on transfer learning," in *Proc. CCF Conf. Big Data*, in Communications in Computer and Information Science, vol. 1120, H. Jin, X. Lin, X. Cheng, X. Shi, N. Xiao, and Y. Huang, Eds., Wuhan, China, 2019, pp. 366–379, doi: 10.1007/978-981-15-1899-7_26.
[56] F. Murtagh and P. Legendre, "Ward's hierarchical agglomerative clustering method: Which algorithms implement ward's criterion?" *J. Classification*, vol. 31, no. 3, pp. 274–295, 2014, doi: 10.1007/s00357-014-9161-z.
[57] T. Lampert, B. Lafabregue, T.-B.-H. Dao, N. Serrette, C. Vrain, and P. Gancarski, "Constrained distance-based clustering for satellite image time-series," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 11, pp. 4606–4621, Nov. 2019, doi: 10.1109/JSTARS.2019.2950406.
[58] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Proc. Int. Conf. Artif. Neural Netw.*, in Lecture Notes in Computer Science, vol. 11141, V. Kurková, Y. Manolopoulos, B. Hammer, L. S. Iliadis, and I. Maglogiannis, Eds., Rhodes, Greece, Oct. 2018, pp. 270–279, doi: 10.1007/978-3-030-01424-7_27.
[59] N. Randriamihamison, N. Vialaneix, and P. Neuvial, "Applicability and interpretability of ward's hierarchical agglomerative clustering with or without contiguity constraints," *J. Classification*, vol. 38, no. 2, pp. 363–389, Jul. 2021, doi: 10.1007/s00357-020-09377-y.
[60] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2670313
[61] G. Vrbancic and V. Podgorelec, "Transfer learning with adaptive fine-tuning," *IEEE Access*, vol. 8, pp. 196197–196211, 2020, doi: 10.1109/ACCESS.2020.3034343.
[62] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms—The numenta anomaly benchmark," in *Proc. IEEE 14th Int. Conf. Mach. Learn. Appl. (ICMLA)*, T. Li *et al.*, Eds., Miami, FL, USA, Dec. 2015, pp. 38–44, doi: 10.1109/ICMLA.2015.141.
[63] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, J. Fürnkranz and T. Joachims, Eds., Haifa, Israel, 2010, pp. 807–814. [Online]. Available: https://icml.cc/Conferences/2010/papers/432.pdf
[64] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds., San Diego, CA, USA, 2015, pp. 1–13.
[65] R. Wu and E. Keogh, "Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress," *IEEE Trans. Knowl. Data Eng.*, early access, Sep. 14, 2021, doi: 10.1109/TKDE.2021.3112126.
[66] K.-H. Lai, D. Zha, J. Xu, Y. Zhao, G. Wang, and X. Hu, "Revisiting time series outlier detection: Definitions and benchmarks," in *Proc. 31st Conf. Neural Inf. Process. Syst. Datasets Benchmarks Track*, 2021, pp. 1–15.

Shenglin Zhang (Member, IEEE) received the B.S. degree in network engineering from the School of Computer Science and Technology, Xidian University, Xi'an, China, in 2012, and the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 2017. He is currently an Associate Professor with the College of Software, Nankai University, Tianjin, China. His current research interests include failure detection, diagnosis, and prediction for service management.

Zhenyu Zhong received the B.S. degree in software engineering from Nankai University, Tianjin, China, in 2020, where he is currently pursuing the M.S. degree with the College of Software. His current research interests include anomaly detection, deep learning, and NLP.

Dongwen Li received the B.S. degree in software engineering from Nankai University, Tianjin, China, in 2019, where she is currently pursuing the Ph.D. degree with the College of Software. Her current research interests include anomaly detection, deep learning, and NLP.

Qiliang Fan received the B.S. degree in software engineering from Nankai University, Tianjin, China, in 2021, where he is currently pursuing the M.S. degree with the College of Software. His current research interests include anomaly detection and deep learning.

Yongqian Sun (Member, IEEE) received the B.S. degree in statistical specialty from Northwestern Polytechnical University, Xi'an, China, in 2012, and the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 2018. He is currently an Assistant Professor with the College of Software, Nankai University, Tianjin, China. His research interests include anomaly detection and root cause localization in service management.

Man Zhu received the B.S. degree in electronic information science and technology from Qufu Normal University, Rizhao, China, in 2019. She is currently pursuing the M.S. degree with the College of Software, Nankai University. Her research interests include anomaly detection and root cause localization.

Yuzhi Zhang received the B.S. and M.S. degrees in computer science from the Department of Computer Science and Technology, Tsinghua University, in 1985 and 1987, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, in 1991. He is currently the Dean of the College of Software, Nankai University, and a Distinguished Professor. His research interests include deep learning and other aspects in artificial intelligence.

Dan Pei (Senior Member, IEEE) received the B.E. and M.S. degrees in computer science from the Department of Computer Science and Technology, Tsinghua University, in 1997 and 2000, respectively, and the Ph.D. degree in computer science from the Department of Computer Science, UCLA, in 2005. He is currently an Associate Professor with the Department of Computer Science and Technology, Tsinghua University. His research interests include network and service management in general. He is an ACM Senior Member.

Jiyan Sun received the B.Eng. degree in computer science and technology from the Beijing University of Posts and Telecommunications, Beijing, China, in 2012, and the Ph.D. degree in signal processing from the University of Chinese Academy of Sciences, Beijing, 2017. She is currently an Assistant Researcher with the Institute of Information Engineering, Chinese Academy of Sciences. Her current research interests include content delivery networks and data center networking.

Yinlong Liu received the B.S. degree from the Department of Electronic and Information Engineering, Hefei University, Hefei, China, in 2004, the M.S. degree from the School of Mechanical Electronic and Information Engineering, China University of Mining and Technology, in 2007, and the Ph.D. degree from the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, in 2011. His research interests include security in wireless networks, network protocol and security, and 5G/B5G.

Hui Yang received the B.S. degree from Carleton University, Canada, in 2011. He is currently a Senior Engineer in artificial intelligence and a Registered Expert of the International Organization for Standardization (ISO). He is also the Founder, the CEO, and the Chairperson of the Board of Directors of Accumulus. His research interests include data analysis and mining.

Yongqiang Zou received the Ph.D. degree in computer system architecture from the Institute of Computing Technology, Chinese Academy of Sciences, in 2010. He is currently a Senior Engineer in artificial intelligence. He is also the Co-Founder and the Chief Technology Officer of Accumulus. His research interests include deep learning, distributed file systems, and intelligent operation and maintenance.