Efficient Multivariate Time Series Anomaly Detection through Transfer Learning for Large-Scale Software Systems

YONGQIAN SUN, MINGHAN LIANG, SHENGLIN ZHANG, ZEYU CHE, ZHIYAO LUO, DONGWEN LI, and YUZHI ZHANG, Nankai University, Tianjin, China
DAN PEI, Tsinghua University, Beijing, China
LEMENG PAN and LIPING HOU, Huawei Technologies Co., Ltd, Shenzhen, China

Timely anomaly detection of multivariate time series (MTS) is of vital importance for managing large-scale software systems. However, many deep learning-based MTS anomaly detection models require long-term MTS training data to achieve optimal performance, which often conflicts with the frequent pattern changes observed in software systems. Moreover, the training overhead of vast MTS in large-scale software systems is unacceptably high. To address these issues, we design *OmniTransfer*, a model-agnostic framework that combines weighted hierarchical agglomerative clustering with an adaptive transfer learning strategy, making many state-of-the-art (SOTA) MTS anomaly detection models efficient and effective. Extensive experiments using real-world data from a large web content service provider and a network operator show that *OmniTransfer* significantly reduces the model initialization time by 46.49% and the training cost by 74.51%, while maintaining high accuracy in detecting anomalies.

CCS Concepts: • **Software and its engineering → Maintaining software**;

Additional Key Words and Phrases: Transfer Learning, Multivariate Time Series, Multivariate Time Series Clustering, Anomaly Detection

ACM Reference format:
Yongqian Sun, Minghan Liang, Shenglin Zhang, Zeyu Che, Zhiyao Luo, Dongwen Li, Yuzhi Zhang, Dan Pei, Lemeng Pan, and Liping Hou. 2025. Efficient Multivariate Time Series Anomaly Detection through Transfer Learning for Large-Scale Software Systems. *ACM Trans. Softw. Eng. Methodol.* 34, 4, Article 89 (April 2025), 25 pages.
https://doi.org/10.1145/3702984

Authors' Contact Information: Yongqian Sun, Nankai University, Tianjin, China; e-mail: sunyongqian@nankai.edu.cn; Minghan Liang, Nankai University, Tianjin, China; e-mail: minghanliang@mail.nankai.edu.cn; Shenglin Zhang (corresponding author), Nankai University, Tianjin, China; e-mail: Shenglin Zhang@nankai.edu.cn; Zeyu Che, Nankai University, Tianjin, China; e-mail: czy@mail.nankai.edu.cn; Zhiyao Luo, Nankai University, Tianjin, China; e-mail: luozhiyao@mail.nankai.edu.cn; Dongwen Li, Nankai University, Tianjin, China; e-mail: lidongwen@mail.nankai.edu.cn; Yuzhi Zhang, Nankai University, Tianjin, China; e-mail: zyz@nankai.edu.cn; Dan Pei, Tsinghua University, Beijing, China; e-mail: peidan@tsinghua.edu.cn; Lemeng Pan, Huawei Technologies Co., Ltd, Shenzhen, China; e-mail: panlemeng@huawei.com; Liping Hou, Huawei Technologies Co., Ltd, Shenzhen, China; e-mail: houliping1@huawei.com.

ACM 1557-7392/2025/4-ART89
https://doi.org/10.1145/3702984

1 Introduction

With the rapid development of the Internet, the scale of software systems has grown exponentially. There are thousands of entities such as containers, virtual machines, and physical machines deployed in **Information Technology (IT)** infrastructure [4, 11, 25, 27, 43, 44]. Anomaly detection is critical to the quality of service management since it helps operators identify anomalous behaviors, improve system stability, and reduce economic losses [27, 34, 41, 47]. Operators configure multiple monitoring metrics for each entity to monitor the running status. These metrics are usually collected continuously at pre-defined intervals. As shown in Figure 1, the monitored metrics of an entity form a **multivariate time series (MTS)**, including system metrics (e.g., CPU load, memory usage, network throughput, and disk I/O) and user-perceived metrics (e.g., average response latency, page visits, and access error rates).

Recently, a series of deep learning-based MTS anomaly detection models have been proposed [2, 7, 9, 22, 23, 33, 38, 53], but they suffer from some limitations. First, they need a long initialization time[1] to perform well. For instance, OmniAnomaly [33] and InterFusion [23] require several weeks of training data. However, operators want to reduce the initialization time when there is a pattern change, such as configuration upgrades or adding new entities. Second, training a model for each entity is impractical as large-scale IT infrastructures have massive entities. Third, the optimal algorithm varies for different scenarios. For example, GDN [9] focuses on the correlation between metrics, while InterFusion [23] also considers temporal dependencies. Therefore, a framework that can effectively reduce initialization time and training overhead and be effective for all models is needed.

There have been some works trying to address the challenges above. **Anomaly Detection in High-Dimensional Time Series with Coarse-to-Fine Model Transfer (CTF)** [35] utilizes clustering and transfer learning to reduce the training overhead of large-scale MTS anomaly detection. Nevertheless, CTF still requires a long model initialization time and only works for the RNN+VAE models [33]. OmniCluster [45] is a model-agnostic framework for large-scale MTS anomaly detection that reduces the training overhead by clustering. However, it is suitable for long-term MTS (i.e., 7 days), resulting in a longer initialization time for anomaly detection. Additionally, CTF and OmniCluster only train the final fine-grained model at the cluster level, which may not apply to all entities within a cluster due to minor shape differences.

Nevertheless, clustering combined with transfer learning is a promising approach to solve these problems [46]. By reducing the number of models through clustering, the training overhead is reduced. Then, fine-tuning the pre-trained model to a new pattern with short-term data can reduce the initialization time. Note that we denote the MTS and models in the source domain as the base MTS and base models, respectively, and the MTS and models in the target domain as the target MTS and target models. However, there are still some challenges when applying clustering and transfer learning.

(1) *High diversity of MTS*. As shown in Figures 1 and 2, the diversity of MTS includes patterns, irregular noise, anomalies, and phase shifts. MTS can be generated by various entities with diverse patterns (i.e., different periodicity, amplitude, trend). Large-scale software systems use different servers to serve users across a wide geographical area, resulting in similar MTS patterns with a time delay. These diversities can affect the distance calculation of MTS and lead to poor clustering performance.

(2) *Aperiodic metrics may reduce the clustering performance.* Figure 1 displays the MTS of different entities. The metrics in the top MTS are with different strengths of periodicity. Many user-perceived metrics and system metrics related to user behavior exhibit periodicity. However, there are also

[1]MTS's model initialization time [28] is defined as the time lag between when the model is launched and when it becomes well trained, mainly influenced by the length of historical data the model needs.

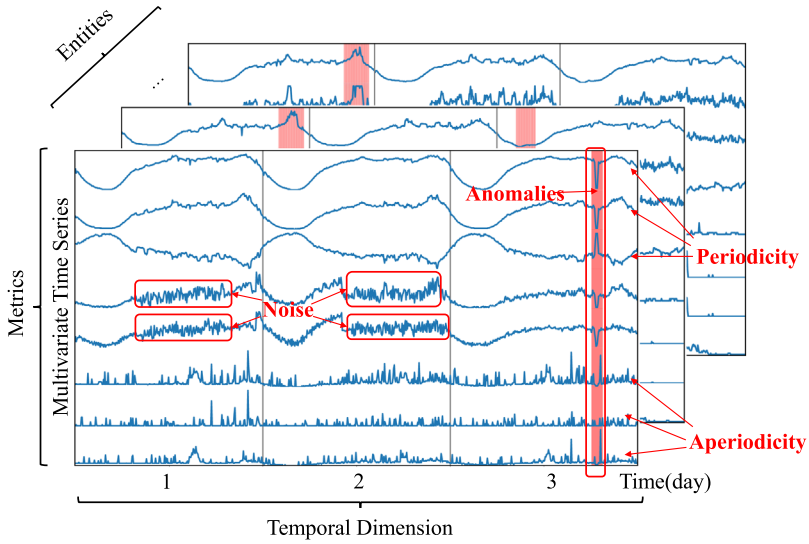

Fig. 1. The MTS of entities in large-scale IT infrastructure.

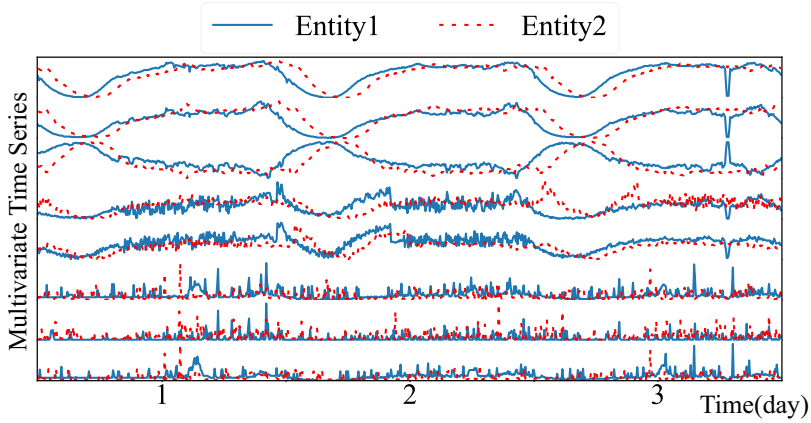

Fig. 2. An example of MTS phase shifts: two MTS are similar in shape but have a time lag.

aperiodic metrics that are unrelated to user behavior. The first three metrics have regular shapes and strong periodicity, which are important for identifying patterns and clustering. The last three metrics do not have regular shapes and contain frequent noise, which will interfere with distance calculation. OmniCluster [45] uses a fixed empirical threshold to remove weak periodicity metrics and keep strong periodicity metrics directly. It may delete metrics with key information and keep metrics with interference. For example, the fourth and fifth metrics in Figure 1 are challenging to define the strength of periodicity they are. It is vital to keep as much information as possible while reducing the interference of aperiodic metrics on clustering.

(3) *Selection of transfer strategy.* There are various strategies for transferring parameters from the base model to the target model. Full parameter transfer and partial parameter transfer strategy are two typical strategies. In most cases, we have the following three observations: (1) The distances between the base and target MTS are various, making the optimal transfer strategy of each target MTS different. (2) The optimal transfer strategies for different models are diverse

for the same dataset. (3) The optimal transfer strategies for different datasets are diverse for the same model. Therefore, we need to use adaptive transfer strategies to achieve better detection performance.

In this article, we propose *OmniTransfer*, an efficient, unsupervised, and model-agnostic framework for MTS anomaly detection. In the offline training stage, *OmniTransfer* uses a **weighted hierarchical agglomerative clustering (W-HAC)** method to cluster the data. It can handle data diversity issues and mitigate the impact of aperiodic metrics. Then, *OmniTransfer* trains a base model for each cluster. When transferring the model to a new pattern MTS, *OmniTransfer* assigns it to the nearest cluster and fine-tunes the base model by an adaptive transfer strategy.

The main contributions of our work are as follows:

(1) We propose *OmniTransfer*, an efficient, unsupervised, and model-agnostic framework for MTS anomaly detection that can significantly reduce the initialization time and the training overhead for large-scale IT infrastructure. *OmniTransfer* uses clustering and transfer learning techniques to transfer the knowledge from well-trained base models to target models. To the best of our knowledge, this is the first model-agnostic framework based on transfer learning for **state-of-the-art (SOTA)** MTS anomaly detection models.
(2) We propose innovative strategies to improve the effectiveness of diversified MTS clustering. We weight metrics based on periodicity to reduce the impact of non-periodic metrics and use phase alignment to eliminate the impact of phase shifts.
(3) We propose an adaptive transfer strategy. It can automatically select either full or partial parameter transfer strategy according to the distance between the target MTS and the base MTS cluster centroid.
(4) We apply *OmniTransfer* on ten SOTA anomaly detection models and conduct experiments with real-world datasets from two top-tier enterprises. Experimental results show that *OmniTransfer* reduces the initialization time by 46.49% and the training cost by 74.51% on average while maintaining high accuracy in detecting anomalies. Furthermore, we make our source code and the labeled datasets publicly available [1] to make it easier for researchers to understand our work.

The rest of this article is organized as follows. Section 2 introduces our motivation for proposing this framework, Section 3 discusses the background, Section 4 discusses the details of the method, Section 5 describes our experimental approach and results, and Section 6 introduces the related work in the same field. Section 7 summarizes lessons learned, future work, and limitations.

2 Motivation

This section elaborates on our motivations by answering the following three questions:

(1) Why do we need to reduce training overhead?
(2) Why do we need to reduce model initialization time?
(3) Why do we need to provide a general framework?

2.1 Why Do We Need to Reduce Training Overhead

Deep learning requires the same distribution between the training and test data, and it is necessary to train a model for each entity because of different data distributions. It will generate a large number of models and a huge training overhead. Table 1 lists the training costs of some MTS anomaly detection models. Such an unacceptable training overhead prevents deep learning-based MTS anomaly detection models from being applied to large-scale software systems.

Table 1. MTS Anomaly Detection Models' Training Overhead

Model	Training Time (1M Entities)
OmniAnomaly [33]	1.57 years
InterFusion [23]	1.41 years
SDFVAE [7]	5.28 weeks
DAGMM [53]	6.09 months
USAD [2]	5.72 weeks
GDN [9]	2.19 weeks
TranAD [38]	4.89 weeks
DOMI [34]	5.15 weeks
SLAVAE [15]	6.07 weeks
MTAD-GAT [49]	3.22 months

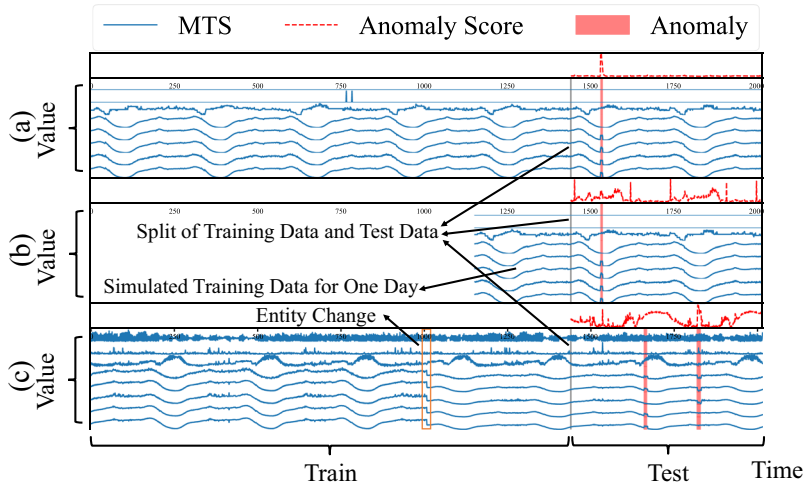

Fig. 3. An example of the impact of addition and change of software systems on model initialization time.

2.2 Why Do We Need to Reduce Model Initialization Time?

Due to the rapid expansion of the Internet, additions and changes of web service entities become more and more frequent [21, 29, 48, 50]. The additions of web service entities generally refer to the horizontal expansion of the service, deploying the original service to a new node, and the monitoring data on the new node lacks the historical training data in a short period. The change of the web service entity includes the release, upgrade, and configuration modification of the service, which will lead to changes in the service running status. Changes, such as less traffic and lower CPU usage due to configuration modifications, are expected.

We use two cases to illustrate the impact of the addition and change of software systems on model initialization time in Figure 3. Figure 3(a) shows a typical entity which uses sufficient data for five days to train the model. For the first case, Figure 3(b) simulates the scenario of insufficient training data when a new entity is added, which starts on the fifth day and has only one day of data for training. Generally, we use F_1 to evaluate the anomaly detection (Section 5 for details). We use OmniAnomaly [33] to get F_1 corresponding to the three types of entities corresponding to Figure 3(a), (b), and (c) on the entire dataset. The F_1 of the entities of type a is 0.99, while the F_1 of the entities of type b is only 0.70. Therefore, the model training is insufficient due to the

Table 2. MTS Anomaly Type

Anomaly Type	Characteristic
Global Anomalies	Exhibiting extreme values compared to all the remaining data.
Contextual Anomalies	Deviating from the neighboring time points.
Pattern Anomalies	Having different basic patterns compared to normal patterns.
Frequency Anomalies	Displaying unusual frequency compared to the overall frequency.
Trend Anomalies	Deviating from the underlying trend of the time series.

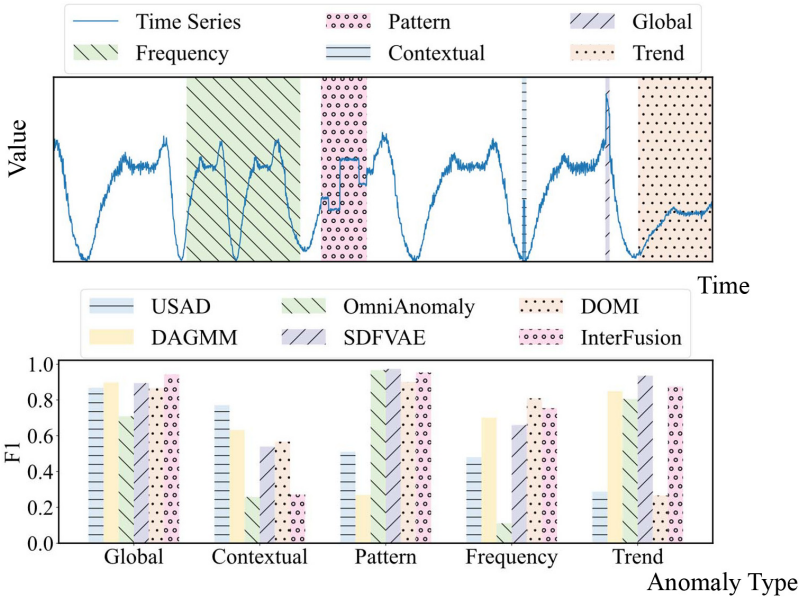

Fig. 4. Five common anomaly types and the result of six SOTA MTS anomaly detection performances for different anomaly types.

lack of training data. For the second case, the entities of type c have a shift change in the training data, resulting in the inconsistency between the distribution of some training data and test data. Correspondingly, the F_1 of this type is 0.31, which is particularly poor.

The above two cases fully illustrate the problem of poor detection performance due to the long model initialization time in the scenarios of addition and change of software systems. Thus proving the necessity of reducing the model initialization time for anomaly detection.

2.3 Why Do We Need to Provide a General Framework?

Different deep models use dedicated designs to detect MTS anomalies in different scenarios. Existing experimental results show that many SOTA models perform differently on different MTS anomaly types. We cite the experimental results of empirical research [10] on many public datasets. The research introduces five anomaly types, shown in Table 2. The upper part of Figure 4 shows a demo of different anomaly types. The lower part of Figure 4 shows the detection performance of six SOTA

models on five anomaly types. The best-performing model is different for each anomaly type. These anomaly types may correspond to different business scenarios. Global anomalies often correspond to obvious business interruptions. For example, excessive traffic causes the service to be temporarily unavailable, often accompanied by an abnormal increase in global resource indicators such as CPU and memory. Trend anomalies may indicate resource configuration changes, modifying the JVM heap and stack configuration, causing the memory size occupied by the new service to steadily increase compared to the occupancy before the change.

Different models have distinct characteristics, making each one suitable for handling different types of anomalies. Therefore, the primary objective of this article is not to investigate the detection capabilities of various anomaly detection models for different types of anomalies. Instead, it aims to propose a general framework that can enhance the transfer learning capabilities of each anomaly detection algorithm.

3 Background

3.1 MTS Anomaly Detection and Clustering

MTS Anomaly Detection. The collected data of each entity forms an MTS with M metrics and N time points as a matrix $X \in R^{M \times N}$. Observing longer data segments reveals discernible specific patterns within MTS. Whenever data deviations from the patterns, it signals an anomaly, potentially indicating a fault in the entity. For each time t, it is necessary to determine whether $X_t \in R^M$ is an anomaly. To quickly catch these anomalies, we usually take a data segment $X_h = (X_{t-W}, X_{t-W+1}, ..., X_t)$ of length W to assist in studying the patterns and further identifying whether X_t is an anomaly [9, 33]. Note that both predicted-based and reconstruction-based methods can be represented by such data segments.

MTS Anomaly Detection Models. There have been many SOTA MTS anomaly detection models proposed, which we can categorize based on their structures. The first type is models consisting of fully connected layers (i.e., Dense layers) [2, 53], typically using a reconstruction-based architecture as depicted in Figure 5(a). The second type is models consisting of specialized layers such as **recurrent neural network (RNN)**, **convolutional neural network (CNN)**, **graph neural network (GNN)**, and attention [7, 9, 15, 23, 33, 34, 38, 49]. These models usually use either a reconstruction-based or predicted-based architecture and are shown in Figure 5(b) and (c). The specialized layers can capture more effective features for anomaly detection. For instance, CNN, attention, and GNN help capture inter-metric dependence, while RNN can capture the temporal dependence of MTS.

MTS Clustering Methods. There have been many studies on MTS clustering, which can be categorized into two types: traditional clustering methods and deep learning-based methods. The first type of method typically employs either the original MTS or low-dimensional representations extracted by traditional machine learning techniques such as **principal component analysis (PCA)** and inverse correlation variance transformation [13, 19, 20, 40]. **Dynamic time warping (DTW)**, **shape-based distance (SBD)**, and Euclidean distance are often used to measure the difference between MTS. However, these methods usually cannot handle the interference of aperiodicity. Meanwhile, DTW and SBD require high computation overhead. The second type of method [35, 45] uses low-dimensional representations extracted by deep learning-based models for clustering. The low-dimensional representations are usually free of noise and can improve clustering efficiency [35, 45]. However, these low-dimensional features lose much information and are usually relevant to subsequent tasks, for example, anomaly detection. Moreover, training deep learning-based models requires significant computing and time resources. To overcome these limitations, we propose a

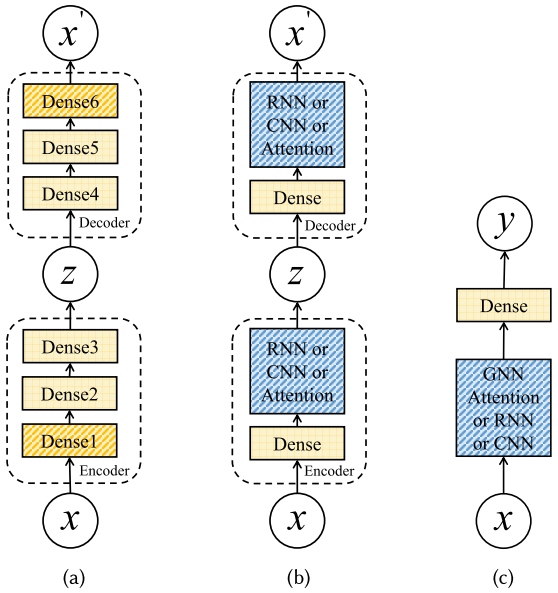

Fig. 5. The neural network architecture of MTS anomaly detection models. (a) Reconstruction-based models with the same modules. (b) Reconstruction-based models with different modules. (c) Prediction-based models with different modules.

task-agnostic clustering method, which ensures the efficiency, effectiveness, and robustness of clustering.

3.2 Transfer Learning

Transfer learning, which focuses on transferring knowledge across domains, is a promising machine learning methodology to solve problems such as insufficient training data and time-consuming training processes [52]. Transfer learning utilizes the knowledge from sufficient source domain data to help the task on the target domain lacking training data. Surveys [31, 52] summarize approaches to transfer learning into four approaches based on "what to transfer." They are the instance-transfer approach, the feature-representation-transfer approach, the parameter-transfer approach, and the relational-knowledge-transfer approach. The instance-transfer approach reuses part of the source domain's data by reweighting or sampling importance in the target domain. The feature-representation-transfer approach improves the performance of the target task by learning a good feature representation from the source domain to the target domain. The parameter-transfer approach aims to share model parameters and prior distributions between the source and the target domains. The relational-knowledge-transfer approach aims to discover the statistical correlation between the source and the target domain data.

This article uses the parameter-transfer approach, combining pre-training and fine-tuning. Transferring the pre-trained model to the target task is usually better than training from scratch [37], which has three main reasons: (1) the performance of the initial model is generally better than that of the randomly initialized model; (2) the learning speed of the fine-tuning is faster than learning from scratch, and the convergence is better; (3) the final performance of the model has better generalizability than training only with target domain data.

However, fully transferring parameters may lead to negative transfer due to the differences in the prior distributions of the source and target domains [5]. To address this, AT-GP [5] and AnoTransfer

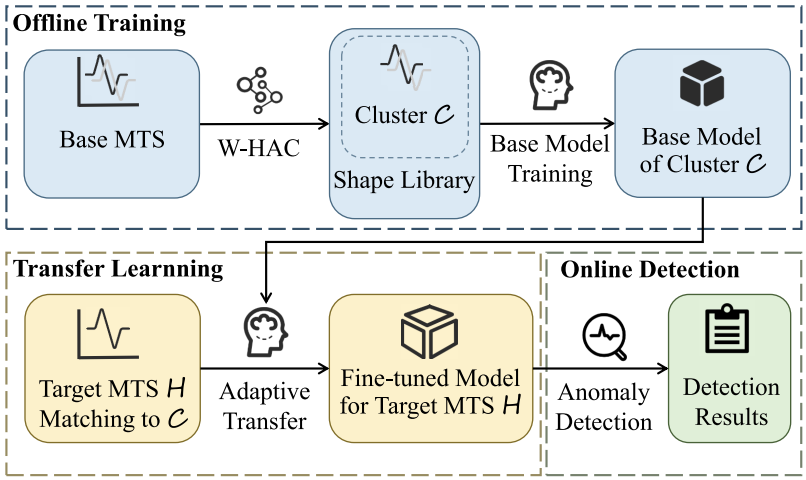

Fig. 6. The overview of *OmniTransfer*.

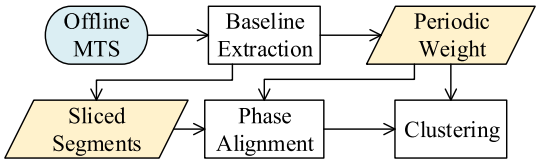

Fig. 7. The overall process of W-HAC.

[46] propose adaptive transfer strategies to automatically select between full parameter transfer and partial parameter transfer strategy. AnoTransfer uses the **normalized cross-correlation (NCC)** to measure the distance among the KPIs. AT-GP formulates the transfer learning problem as a unified Gaussian Process model. They both avoid negative transfer during the transfer learning and achieve better generalizability.

4 Approach

4.1 Overview

We propose a model-agnostic framework, named *OmniTransfer*, to reduce initialization time and training overhead of MTS anomaly detection. Figure 6 shows the overview of *OmniTransfer*, which includes three main stages: offline training, transfer learning, and online detection.

The offline training stage comprises two steps: W-HAC and base model training. Figure 7 illustrates the process of W-HAC. To reduce interference from aperiodic metrics, we weigh the contribution of metrics to clustering based on their strength of periodicity. Besides, we address the problem of the MTS phase shifts. Thus, W-HAC can group MTS with similar shapes, addressing the first and second challenges. In the base model training stage, *OmniTransfer* trains a base model that can be used for transfer learning by using several MTS segments near the cluster centroid.

The target MTS undergoes transfer learning and online detection stages sequentially. First, we match the short-term data of the target MTS to an appropriate cluster and then use an adaptive transfer strategy to fine-tune the corresponding base model. The adaptive transfer strategy selects the best transfer strategy based on the distance between the target MTS and its corresponding

cluster centroid, which solves the third challenge. Finally, in the online detection stage, we use the fine-tuned model to detect anomalies in the target MTS.

4.2 Pre-Processing

Data pre-processing is crucial for offline training, transfer learning, and online detection stages since it is hard to guarantee that all monitoring data are ideally collected in large-scale IT infrastructure. According to the previous experience [46], the proportion of missing points is typically less than 5%. We can fill in these missing points directly by utilizing linear interpolation. Another widely used pre-processing step for time series is standardization, which is useful for eliminating the impact of amplitude by scaling the data to a standard normal distribution. The process of standardization is given by (1),

$$\mathbf{X}'^{j} = \frac{\mathbf{X}^{j} - mean(\mathbf{X}^{j})}{std(\mathbf{X}^{j})}, \tag{1}$$

where $X^j \in R^N$ is the jth metric after filling in the missing value, and $\mathbf{X}'^{j} \in R^N$ is the jth metric after standardization.

4.3 Offline Training

4.3.1 W-HAC. The W-HAC (illustrated in Figure 7) aims to reduce the diversity of MTS and thus lower the training overhead of anomaly detection models. The specific steps of W-HAC are as follows:

Baseline Extraction. Noise and anomalies can significantly impact the normal pattern of MTS and increase the diversity of MTS patterns, as mentioned in the first challenge. To address this issue, we extract the baselines (normal patterns) of MTS by removing extreme values and applying a moving average. Extreme values are more likely to be anomalies and their ratio is often less than 5% [24, 45, 46]. Therefore, W-HAC removes the top 5% data that deviates from the mean value and then uses linear interpolation to fill the vacancies. Then, W-HAC applies the moving average to reduce the impact of noise.

Periodic Weights. To determine the strength of periodicity of each metric in MTS, we use the **cumulative mean normalized difference (*CMND*)** [8], which is an improved version of the autocorrelation-based approach and well suited for long-term data. *CMND* is given by (2), where τ is an empirical candidate periodicity value, such as 1 hour, 1 day, 1 week, or 1 month.

$$\begin{aligned} d(\tau) &= \sum_{i=1}^{N-\tau} (\mathbf{u}_i - \mathbf{u}_{i+\tau})^2 \\ CMND(\tau) &= \frac{d(\tau)}{[(1/\tau)\sum_{j=1}^{\tau} d(j)]} \end{aligned}. \tag{2}$$

For each metric in the MTS, we calculate the *CMND* and then average them across the entity dimension to obtain $\mathbf{P} \in R^M$, where M is the number of metrics in the MTS. The smaller $\mathbf{P}^j$, the stronger the periodicity of the jth metric. We aim to assign high weights to strong periodic metrics in clustering. Thus, we compute the periodic weight $\mathbf{PW} \in \mathbb{R}^M$ by $\mathbf{PW} = \mathbf{P}^{-\alpha}$, where α is a hyper-parameter. A larger value of α leads to a greater weight difference between metrics with different levels of periodicity.

Segmentation of MTS. After computing the baseline and periodic weights, we slice MTS into short-term segments, denoted as $\mathbf{MTS}_{seg} \in R^{M \times n}$, that match the length of the target MTS. Here, n represents the time points after segmentation.

Instead of MTS entities, we use MTS segments as input for clustering and transfer learning to reduce model initialization time and training cost. The use of shorter MTS segments allows for the selection of suitable clusters corresponding to the base model. When performing anomaly detection for a new MTS data segment, the models can be fine-tuned well with less data. Moreover, using complete entities for transfer learning requires longer data for cluster matching and model fine-tuning. Additionally, the entity data need to be as consistent in length as possible. Clustering entities of different lengths tend to be less accurate.

Phase Alignment. We then combine **PW** to align the phase shift because discussing the phase shift for aperiodic metrics is less meaningful.

First, we get the pivot **PVT** of the entire offline segments $\mathcal{D}$ according to (3). The weighted Euclidean distance between two $\mathbf{MTS}_{seg}$ can be calculated by (4).

$$\mathbf{PVT} = \arg\min_{\mathbf{A}\in\mathcal{D}} \sum_{\mathbf{B}\in\mathcal{D}} Euc_w(\mathbf{A},\mathbf{B}) \tag{3}$$

$$Euc_w(\mathbf{A},\mathbf{B}) = (\mathbf{A}-\mathbf{B})^2 \times \mathbf{PW}. \tag{4}$$

Next, we use **weighted NCC (NCC_w)** to estimate the best phase shift for all $\mathbf{MTS}_{seg}$ to align to **PVT**. $s \in [-n+1, n-1]$ denotes the possible phase shifts. To retain short-term information, we use (5) to wrap round MTS.

$$\begin{aligned} \mathbf{A}(s) &= (\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_n) \\ \mathbf{B}(s) &= \begin{cases} (\mathbf{B}_{n-s+1}, \ldots, \mathbf{B}_n, \mathbf{B}_1, \ldots, \mathbf{B}_{n-s}) & s \geq 0, \\ (\mathbf{B}_{-s+1}, \ldots, \mathbf{B}_n, \mathbf{B}_1, \ldots, \mathbf{B}_{-s}) & s < 0. \end{cases} \end{aligned} \tag{5}$$

NCC_w reaches the maximum value when s is close to the real phase shift, which is given by (6).

$$\begin{aligned} CC_w(\mathbf{A},\mathbf{B},s,j) &= \sum_{i=1}^{n} \mathbf{A}(s)_i^j \cdot \mathbf{B}(s)_i^j \cdot \mathbf{PW}^j \\ NCC_w(\mathbf{A},\mathbf{B},s) &= \sum_{j=1}^{M} \frac{CC_w(\mathbf{A},\mathbf{B},s,j)}{||\mathbf{A}(s)^j||_2 \cdot ||\mathbf{B}(s)^j||_2}. \end{aligned} \tag{6}$$

The best phase shift s^* obtained by (7).

$$s^* = \arg\max_{s\in[-n+1,n-1]} NCC_w(\mathbf{PVT}, \mathbf{MTS}_{seg}, s). \tag{7}$$

Finally, we align the phase shift s^* of $\mathbf{MTS}_{seg}$ to get $\mathbf{MTS}'_{seg}$.

Clustering. OmniTransfer gets the clustering result using **hierarchical agglomerative clustering (HAC)** and the weighted Euclidean distance. HAC with average linkage is adopted for the following reasons. (1) The HAC algorithm is robust to the extreme value because it clusters on the rank of distances rather than the value. (2) Each data in the cluster have the same effect on the distance measure, making the distance measure transitive. After clustering, several segments near the cluster centroid are saved for base model training.

4.3.2 Base Model Training. The VAE-based algorithms [7, 23, 33] model the relationship between the latent variable z and the observed variable x. They typically train their models by optimizing the evidence lower bound described in (8), which is comprised of a reconstruction probability and a regularization term. p_θ is a generative model that represents the real posterior of the data, while q_ϕ is an inference model aiming to estimate the posterior. The D_{KL} term represents the Kullback-Leibler divergence [18]. On the other hand, AE-based and prediction-based models

[2, 9, 38, 53] focus on reconstructing or predicting the target. These models train by minimizing the difference between the target and output in (9).

$$\mathcal{L}_1 = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}[q_\phi(z|x)||p_\theta(z)] \tag{8}$$

$$\mathcal{L}_2 = \mathbb{MSE}(target - output). \tag{9}$$

4.4 Transfer Learning

Transfer Preparations. To train the target model for each target MTS, *OmniTransfer* utilizes a base model E, which is selected based on the cluster centroid's proximity to the target short-term data $\mathbf{H} \in R^{M \times n}$. First, we perform baseline extraction and phase alignment to get $\mathbf{H}^{'}$. Then, we calculate the distance between $\mathbf{H}^{'}$ and the centroid of each cluster and select the closest one and its corresponding base model for transfer learning. We use $\mathbf{H}$ to fine-tune the base model.

Adaptive Transfer Strategy. We propose an adaptive transfer strategy that automatically selects whether to transfer full parameters or partial parameters for each target MTS. When the target MTS and the nearest cluster centroid are relatively similar, we use the full parameter transfer strategy and fine-tune the entire base model's parameters directly. Otherwise, we employ the partial parameter transfer strategy. Specifically, we initialize a target model with random parameters and load part of the base model's parameters into the target model. First, we update the remaining parameters while keeping the transferred parameters fixed. Then we fine-tune all of the parameters of the target model.

Distance Measurement. We use a distance measurement to help decide which transfer strategy to select for each target MTS. The anomaly score measures the deviation between the target data and the normal pattern learned by the base model. We use the summation across all time points anomaly scores as the distance score. To avoid the impact of anomalies and noise in the data, we remove the top 5% of the anomaly scores. The distance score is defined as (10), where $AnomalyScore_E^{'}$ is obtained by removing extreme values from either (11) or (12).

$$DiffScore_E(\mathbf{H}) = sum(AnomalyScore_E^{'}(\mathbf{H})). \tag{10}$$

The threshold value β for $DiffScore$ is usually determined by experienced operators or initialized by referring to some entities in the dataset. Empirically, applying the initial β is sufficient to achieve good results. As the data volume increases, the optimal value for β can be updated to further enhance the detection performance.

Transfer Layer Selection. We adopt the partial parameter transfer strategy when there is a significant difference between the target MTS and its corresponding base model. We select specific layers based on the models' capabilities and characteristics for transferring. As mentioned in Section 3.1, these SOTA MTS anomaly detection models fall into two categories based on their structures. For the former type, their outer layers focus on more general tasks and capture more generic features [3, 36, 42], while the inner layers are designed to capture more task-specific features [12, 39]. For the latter, the specialized layers (e.g., RNN, CNN, attention, and GNN) capture more generic features, while the fully connected layers focus more on specific tasks [6, 16, 26, 32, 35]. It is recommended to transfer the parameters of the outer layers or the specialized layers when adopting the partial parameter transfer strategy, as they learn generic features that are often not specific to a particular task.

4.5 Online Detection

We use the fine-tuned model for online detection. For the VAE-based models, their anomaly score corresponds to the negative reconstruction probability, which is given by (11). $\log p_\theta(x|z)$ denotes the reconstruction probability of each observed variable x. The smaller the reconstruction

probability, the greater the probability that this data point is an anomaly. For the AE-based models and prediction-based models, we calculate the anomaly scores according to (12), which measures the difference between the target and the output. A greater difference indicates a higher probability that the data point is an anomaly.

In addition, determining the anomaly score threshold is crucial to identify the anomaly points. To obtain the best results, we use grid search to select the optimal threshold from the available range during evaluation.

$$AnomalyScore_1 = -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] \tag{11}$$

$$AnomalyScore_2 = \mathbb{MSE}(target - output). \tag{12}$$

In previous grid search selection of the optimal threshold selection, the selection was based on the test set. We also prepared validation sets, but in a previous work we used a method called the "Tree Based Pipeline Optimization Tool" to optimize the machine learning pipeline, this article used a test set to select the best threshold, so this article also used a test set based approach to select the best threshold.

5 Evaluation

In this section, we first introduce the experimental setup, including dataset, experiment environment, evaluation metrics, and hyper-parameters of *OmniTransfer*. Then, we conduct extensive experiments to evaluate the performance of *OmniTransfer* and answer the following **research questions (RQs)**:

RQ1. How does the effectiveness and efficiency of *OmniTransfer* compare to baseline methods?
RQ2. How much initialization time can *OmniTransfer* reduce compared to non-transfer learning methods?
RQ3. How much do the key techniques contribute to the overall performance?
RQ4. How well does the W-HAC perform compared to other clustering methods?
RQ5. How does the transfer strategy threshold influence the performance?

5.1 Experimental Setup

Dataset and Environment. In this work, we use two MTS datasets, Dataset1 is derived from the operating systems and service data of a multitude of servers, which monitors the system software data and application performance data when the machines provide services to the users. Dataset2 encompasses software system data from wireless base stations of one of the world's leading ISPs. It provides a comprehensive reflection of the monitoring data, capturing both user behavior and service status, offering valuable insights into the performance and operational dynamics of the wireless communication infrastructure.

More specific details are shown in Table 3. We do not use public datasets (e.g., SWaT and WADI [30], SMD [33], SMAP and MSL [17]), mainly because the number of entities is too small (i.e., less than 55 entities).

Please note that we only choose 400 entities from millions for evaluation since the labeling is time-consuming. In real-world scenarios, additions or upgrades are relatively rare occurrences. To simulate MTS pattern changes, we employ different entities from the original dataset. To be more specific, we randomly choose 50% of the entities for training models offline, while the remaining 50% represent newly added entities used for transfer learning and online detection. The online data are labeled by experienced operators based on real service faults using the labeling tools provided by CTF [35]. The source code of *OmniTransfer* and the datasets are publicly available in [1]. All experiments are run on a server with two 16C32T Intel Xeon Gold 5218 CPU @ 2.30 GHz, one NVIDIA Tesla V100S, and 192 GB RAM.

Table 3. Dataset Details

	Dataset1	Dataset2
Entity type	Web server	Wireless base station
Number of entities	400	400
Number of metrics	19	25
Base model training data duration	7 days	14 days
Transfer training data duration	1 day	1 day
Test data duration	2 days	7 days
Anomaly proportion	5.52%	5.18%

Evaluation Metrics. OmniTransfer outputs an anomaly score for each point and determines whether it is an anomaly by a threshold. Thus, MTS anomaly detection can be regarded as a binary classification problem. We use the F_1 to evaluate the effectiveness, which is given by (13). TP represents true positives, FP represents false positives, and FN represents false negatives. The F_1 of each dataset is obtained using the micro-average method. By enumerating all possible thresholds, we obtain the best F_1 for each model, denoted by F_1^*. Additionally, we record the time required for model training to evaluate efficiency.

$$\begin{aligned} Precision &= \frac{TP}{TP + FP} \\ Recall &= \frac{TP}{TP + FN} \\ F_1 score &= 2 \times \frac{Precision \times Recall}{Precision + Recall}. \end{aligned} \tag{13}$$

Hyper-Parameters. We use the best empirical values for most parameters based on experimental results. Specifically, We set the sliding window length for the moving average to 12 and 4 for the two datasets, respectively. The exponents α for the periodic weights applied to different metrics during clustering are 1 for the two datasets. We use 5 and 20 segments closest to the centroid for each cluster to train the base models for the two datasets, respectively. For all MTS, we slice them using a sliding window with a length of 60. The epoch and learning rate of each base model training, full-parameters transfer strategy fine-tuning, and partial-parameters transfer strategy fine-tuning are presented in Table 4. The best threshold β is shown in Table 4.

Point-Adjust (PA) *Strategy.* In our experimental evaluation, we employed the PA strategy, a widely recognized protocol in time series anomaly detection that adjusts the anomaly predictions by considering the entire contiguous segment as detected if at least one point within it exceeds the anomaly threshold. This method is particularly effective in scenarios where the detection of any anomaly within a period is sufficient to trigger necessary actions, thereby providing a practical approach to assess the performance of our anomaly detection models.

Validation Set. In our experimental evaluation, we used the validation set to select the threshold. The first half of the labeled test set is used as the verification set to select the threshold value of the abnormal score. The F1 score of the second half of the test set is calculated by using the calculated threshold value, and the result is added to Table 6 in Section 5.2.

5.2 *OmniTransfer* vs. Baseline Models

To demonstrate the effectiveness and efficiency of *OmniTransfer*, we compare it with OmniCluster [45], one model/entity, CTF [35], JumpStarter [28], and Uni-AD [14]. In addition, we have incorporated one of the most representative pre-training models, "One Fits All" [51], given that time series

Table 4. Hyper-Parameters Settings

Model	Dataset1							Dataset2						
	$epoch_b$	lr_b	$epoch_f$	lr_f	$epoch_p$	lr_p	β	$epoch_b$	lr_b	$epoch_f$	lr_f	$epoch_p$	lr_p	β
OmniAnomaly	50	0.001	10	0.0005	10	0.001	868	50	0.001	10	0.0005	10	0.001	107
InterFusion	10	0.0005	10	0.0003	20	0.0005	807	10	0.0005	10	0.0003	20	0.0005	1,039
SDFVAE	100	0.001	10	0.001	20	0.001	2,430	100	0.002	20	0.0005	20	0.0005	364
DAGMM	500	0.001	20	0.002	50	0.006	7,157	500	0.001	20	0.005	50	0.003	9,917
USAD	100	0.001	5	0.0001	24	0.001	223	100	0.001	5	0.0002	10	0.001	132
GDN	50	0.005	10	0.0005	30	0.005	2,195	50	0.005	10	0.0005	20	0.005	1,152
TranAD	100	0.0005	10	0.0005	20	0.005	199	100	0.0005	10	0.0001	20	0.0001	24
DOMI	100	0.001	10	0.001	20	0.0005	849	100	0.002	10	0.0005	20	0.001	126
SLAVAE	100	0.001	20	0.0005	10	0.001	2,164	100	0.0001	20	0.0005	10	0.0005	251
MTAD-GAT	50	0.001	30	0.001	40	0.001	633	30	0.001	10	0.001	40	0.001	247

$epoch_b$, $epoch_f$ and $epoch_p$ denote the epochs of base model training, full-parameters transfer strategy fine-tuning, and partial-parameters transfer strategy fine-tuning, respectively. lr_b, lr_f and lr_p denote the learning rate similarly.

pre-training models have been extensively studied recently. One Fits All model avoids changing the self-attention and feedforward layers of residual blocks in the pre-training language or image model, and can produce equivalent or the most advanced performance in all major time series analysis tasks. The details are as follows: (1) OmniCluster is a model-agnostic framework for MTS anomaly detection. (2) One model/entity involves training a separate model for each MTS. (3) CTF is a transfer-based framework to achieve scalable anomaly detection. (4) JumpStarter is an MTS anomaly detection model that jump-starts quickly with a short initialization time. (5) Uni-AD is a transformer-based model that works well for model sharing. *OmniTransfer*, OmniCluster, and one model/entity are model-agnostic training frameworks or strategies that can be combined with various deep anomaly detection models.

To demonstrate the advantages of the PA strategy, we also compared it with the case where the PA strategy was not used.

We combine these frameworks/methods with 10 typical unsupervised MTS anomaly detection methods: OmniAnomaly, InterFusion, SDFVAE, DAGMM, USAD, GDN, TranAD, DOMI, SLAVAE, and MTAD-GAT. These models focus on different challenges in MTS anomaly detection and have different structures. Table 5 shows the structure and characteristics of these selected models. The results of these methods are presented at the top of Table 6. CTF is designed specifically for the RNN+VAE model, JumpStarter is not based on deep learning and cannot be combined with *OmniTransfer*, and Uni-AD designed a special model based on the transformer. The results of these three baselines are shown at the bottom of Table 6. *OmniTransfer* outperforms all baselines in effectiveness and is more efficient than all baseline models except for OmniCluster. We will try to analyze the reasons for this result in detail.

Compare with OmniCluster. On Dataset1, *OmniTransfer* outperforms OmniCluster by 14.34% to 88.06%, while on Dataset2, *OmniTransfer* outperforms OmniCluster by 8.84% to 94.46%. We attribute this to *OmniTransfer* improving the clustering method and *OmniTransfer* training a better model for each MTS. *OmniTransfer* applies periodic weighting to the metrics instead of removing some metrics directly, which allows for a more comprehensive use of information. In contrast, OmniCluster compresses MTS in the temporal dimension and removes some metrics, resulting in a loss of both shape and metric information. *OmniTransfer* uses transfer learning to train a suitable model for each MTS, whereas OmniCluster trains a base model for each cluster.

The training time of *OmniTransfer* is 29.94% and 52.24% higher than OmniCluster on two datasets. Because OmniCluster only trains base models without fine-tuning. Nevertheless, effectiveness is usually more important than efficiency in practice, making *OmniTransfer* a superior solution to OmniCluster.

Table 5. Selected Anomaly Detection Models

Model	Structure	Characteristics
OmniAnomaly [33]	RNN+VAE	For the first time, handling temporal dependence and stochasticity of MTS and learning robust representation.
InterFusion [23]	1D-CNN+RNN+HVAE	Novelly employing HVAE to obtain inter-metric embeddings and temporal embeddings.
SDFVAE [7]	2D-CNN+RNN+VAE	Making use of time invariance in MTS to enhance the robustness and noise-resistance.
DAGMM [53]	AE+GMM	Using joint optimization to address the decoupling problem in the model learning.
USAD [2]	AE+GAN	The combined use of AE and GAN results in a more stable and faster model training process.
GDN [9]	GNN+Attention	GNN can accurately capture the correlations among metrics.
TranAD [38]	AE+Attention+GAN	Enabling powerful multi-modal feature extraction and adversarial training improves stability.
DOMI [34]	1D-CNN+GMM+VAE	Learning potential representations of machine instances to capture their normal patterns.
SLAVAE [15]	1D-CNN+RNN+VAE	Active learning is employed to update the online model with a small number of uncertain samples.
MTAD-GAT [49]	GNN+Attention	Leveraging two parallel graph attention layers to learn the relationships between different metrics dynamically.

Table 6. The Overall Performance of *OmniTransfer* Compared to Baseline Models

Model	Dataset1											
	OmniTransfer				OmniCluster				One Model/Entity			
	F_1^*	Time (s)	Validation F_1^*	Not PA F_1^*	F_1^*	Time (s)	Validation F_1^*	Not PA F_1^*	F_1^*	Time (s)	Validation F_1^*	Not PA F_1^*
OmniAnomaly	**0.9721**	1,212.99	0.9452	0.6795	0.5169	560.47	0.4751	0.5758	0.7000	9,888.25	0.6369	0.4933
InterFusion	**0.9047**	1,585.63	0.8892	0.6706	0.5830	566.56	0.5209	0.5609	0.4769	8,884.94	0.4286	0.76
SDFVAE	**0.8512**	209.73	0.8426	0.6447	0.4922	178.02	0.4155	0.4916	0.6055	638.93	0.5217	0.8445
DAGMM	**0.8738**	244.48	0.8521	0.6764	0.7104	137.37	0.5639	0.5653	0.8245	2,947.47	0.7642	0.7377
USAD	**0.8539**	80.16	0.8318	0.693	0.7468	109.04	0.7084	0.6334	0.7875	691.77	0.7184	0.602
GDN	**0.8037**	54.55	0.7756	0.481	0.6806	42.81	0.6129	0.4253	0.7405	265.27	0.6872	0.5189
TranAD	**0.9714**	114.53	0.9208	0.909	0.7797	102.10	0.7084	0.7389	0.8538	591.67	0.8144	0.9388
DOMI	**0.8849**	156.58	0.8529	0.6215	0.6418	119.56	0.5421	0.5542	0.7138	623.65	0.5871	0.6473
SLAVAE	**0.8417**	142.54	0.7122	0.6641	0.4831	101.34	0.4508	0.4539	0.5817	603.45	0.4428	0.5514
MTAD-GAT	**0.9414**	1,149.95	0.9098	0.6417	0.6466	305.06	0.6072	0.4792	0.9064	1,666.67	0.8413	0.6837
JumpStarter	0.4211	4,786.67	0.5227	0.458	-	-	-	-	-	-	-	-
CTF	0.8661	4,965.61	0.3456	0.7257	-	-	-	-	-	-	-	-
Uni-AD	0.6232	119.95	0.5489	0.5759	-	-	-	-	-	-	-	-
One Fits All	0.9218	5,216.18	0.9127	0.7642	-	-	-	-	-	-	-	-

Model	Dataset2											
	OmniTransfer				OmniCluster				One Model/Entity			
	F_1^*	Time (s)	Validation F_1^*	Not PA F_1^*	F_1^*	Time (s)	Validation F_1^*	Not PA F_1^*	F_1^*	Time (s)	Validation F_1^*	Not PA F_1^*
OmniAnomaly	**0.974**	1,430.14	0.9489	0.9106	0.7885	522.63	0.7109	0.5179	0.6316	7,791.65	0.5892	0.6446
InterFusion	**0.9235**	1,131.33	0.8962	0.8085	0.6756	479.01	0.6013	0.4819	0.4639	5,870.73	0.3872	0.5621
SDFVAE	**0.8673**	572.95	0.8127	0.7799	0.446	230.75	0.4321	0.5232	0.819	1,402.87	0.7356	0.8453
DAGMM	**0.9439**	271.29	0.8907	0.851	0.8048	133.57	0.7519	0.7066	0.9047	2,923.78	0.8265	0.7658
USAD	**0.9355**	138.39	0.8839	0.8334	0.7138	93.01	0.6692	0.5664	0.8514	665.19	0.7691	0.8753
GDN	**0.9525**	46.03	0.9088	0.7835	0.7503	17.15	0.6873	0.5429	0.9382	301.17	0.8819	0.7619
TranAD	**0.9323**	201.93	0.8873	0.8467	0.8566	82.40	0.7863	0.8196	0.5273	704.29	0.479	0.3334
DOMI	**0.9316**	309.25	0.7429	0.7537	0.8136	87.35	0.5421	0.6931	0.8426	1,059.76	0.5871	0.7431
SLAVAE	**0.8589**	465.77	0.7122	0.6308	0.8136	216.20	0.4508	0.6109	0.8025	1,304.03	0.4428	0.6706
MTAD-GAT	**0.9757**	262.68	0.9133	0.7814	0.5338	204.87	0.4802	0.4672	0.7682	506.35	0.6945	0.7439
JumpStarter	0.649	5,359.1	0.4852	0.5227	-	-	-	-	-	-	-	-
CTF	0.8788	6,187.86	0.4454	0.7645	-	-	-	-	-	-	-	-
Uni-AD	0.5978	23.30	0.5196	0.5931	-	-	-	-	-	-	-	-
One Fits All	0.9167	5,971.93	0.8792	0.7831	-	-	-	-	-	-	-	-

The bold values indicate the maximum value in a row in a dataset.

Comparison with One Model/Entity. In terms of F_1, *OmniTransfer* achieves an average improvement of 27.84% and 31.67% on the two datasets, respectively. When using a single entity model, ideally, with sufficient training data, the detection results are similar to those of the migration base model. One model/entity uses only short-term MTS for training, which is insufficient for deep learning-based models. However, in most cases, when there is insufficient training data for online entities, if a single entity model is not used based on the migration base model, the model training will be insufficient due to insufficient training data, resulting in poor detection results. Moreover, training the model from scratch usually takes longer to converge. As the amount of data increases, the training overhead increases significantly. Furthermore, *OmniTransfer* reduces the training overhead by 75.95% and 73.07%. After clustering, the number of basic models is much smaller than the number of entities. Fine-tuning is performed on the basic model, the model converges faster, the number of training rounds required is smaller, and the overall training cost is lower. Therefore, the performance and efficiency of one model/entity strategy are unsatisfactory. In contrast, *OmniTransfer* performs better by maximizing the use of the base MTS to train the base model. The overall training overhead of *OmniTransfer* benefits from only a small number of base models that need to be trained and the base models help accelerate the convergence of the target model training.

Comparison with Not PA. The results show that *OmniTransfer* still outperforms most of the other baseline models. However, every model' F_1^* is notably diminished without the application of the PA strategy, which may be due to insufficient accuracy in data collection or in the precision of anomaly labeling. This could be the reason why other baseline studies all employ the PA strategy.

Comparison with CTF. The CTF is specifically designed for RNN+VAE-based models, particularly for OmniAnomaly. Therefore, we only compare the performance of *OmniTransfer*+OmniAnomaly with CTF. The F_1 of *OmniTransfer* + OmniAnomaly is approximately 10% higher than CTF. CTF produces a fine-tuned model at the cluster level, which cannot be deployed perfectly to each MTS. The training time of CTF is more than four times that of *OmniTransfer*+OmniAnomaly on two datasets. This is because CTF fine-tunes cluster-level models based on a dataset-level pre-trained model. As the difference between the source domain and the target domain of CTF is significant, it requires more MTS and training epochs during fine-tuning.

Comparison with JumpStarter. JumpStarter successfully reduces model initialization time by sampling from the data and reconstructing the data for anomaly detection based on the sample. However, its F_1 is significantly lower and the training time is much longer compared to *OmniTransfer*. JumpStarter uses only uses short-term data to sample and reconstruct the normal value, which is usually sufficient. And the outlier-resistant sampling method may not always successfully remove anomaly points in highly volatile metrics, limiting the performance of JumpStarter. Additionally, the complicated sampling process in JumpStarter increases the training time seriously.

Comparison with Uni-AD. Uni-AD employs model sharing to address the challenges posed by large-scale, diverse, and dynamic MTS. Based on transformer encoder layers, Uni-AD can model diverse patterns for different monitored entities. On Dataset1, the training time of Uni-AD is similar to *OmniTransfer* and has less training time on Dataset2 because it uses model sharing to reduce the number of models and the model structure of the transformer is light-weighted. However, its F_1 is significantly lower compared to *OmniTransfer*. Uni-AD focuses on a large amount of data with the same pattern and performs poorly when the patterns among different entities diverge.

Comparison with One Fits All. When compared with the *OmniTransfer* versions of 10 models, the F1* of One Fits All is relatively balanced, ranking fourth on Dataset1 and ninth on Dataset2. Additionally, in terms of efficiency, *OmniTransfer* performs much better than One Fits All. The training time for One Fits All is more than 10 times longer than the average training time of *OmniTransfer* on both Dataset1 and Dataset2. However, as a single model, One Fits All has a higher F1* than the most single models, indicating its strong general applicability.

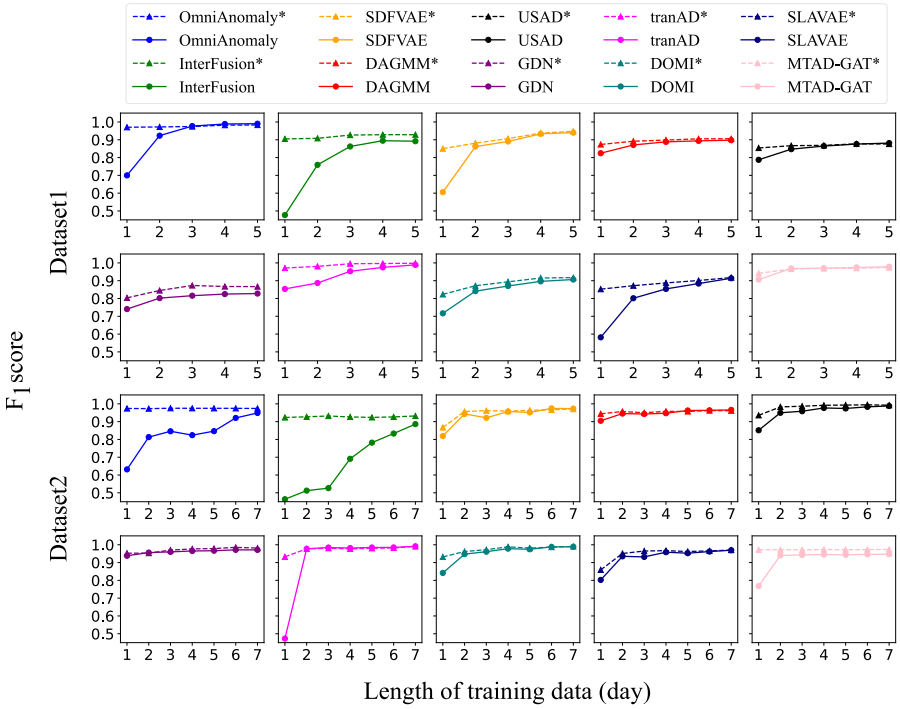

Fig. 8. The performance of *OmniTransfer* and one model/entity with different initialization time. "*" denotes the corresponding result of combining *OmniTransfer*, and without "*" denotes the result of one model/entity strategy.

Comparison with Validation Set. It is evident that the *OmniTransfer* version continues to outperform the other models, even though the validation F1* scores for each model are slightly lower than the F1* scores.

5.3 Effect on Reducing Model Initialization Time

In this section, we conduct experiments on 10 anomaly detection models to verify the effect of *OmniTransfer* in reducing model initialization time. We increase the initialization time for the two datasets from 1 day to 5 days and 1 day to 7 days. Figure 8 demonstrates that *OmniTransfer* outperforms one model/entity by 16.53% and 21.48% with 1 day and 2 days of training data on average. *OmniTransfer* using 2 days of training data performs almost the same as one model/entity using all training data. This highlights its ability to significantly reduce model initialization time. Specifically, the pre-training knowledge of the basic model based on offline data is used, and only a small amount of online data is needed to achieve good detection results, reducing the model initialization time. Moreover, the performance of both *OmniTransfer* and one model/entity improves as the initialization time increases. However, for *OmniTransfer*, the performance becomes stable after using less than 2 days of training data, while for one model/entity, the performance of most models is unsatisfactory with less than 3 days of training data.

5.4 Ablation Experiment

To demonstrate the effect of five key technologies in *OmniTransfer*: (1) clustering; (2) weighting metrics; (3) aligning phases; (4) transfer learning; and (5) adaptive transfer strategy, we reconfigure

Table 7. Ablation Experiment

Model	Dataset1								Dataset2							
	OmniTransfer	C1	C2	C3	C4	C5	C6	C7	*OmniTransfer*	C1	C2	C3	C4	C5	C6	C7
OmniAnomaly	**0.9721**	0.6452	0.8239	0.8018	0.694	0.775	0.9675	0.9675	**0.974**	0.6371	0.8092	0.9297	0.703	0.9194	0.9739	**0.974**
InterFusion	**0.9047**	0.566	0.6963	0.7686	0.6944	0.8115	0.9037	0.9037	**0.9235**	0.7184	0.6128	0.8564	0.6702	0.8818	0.8948	0.9061
SDFVAE	**0.8512**	0.6513	0.7111	0.7825	0.635	0.7163	0.8463	0.8485	**0.8673**	0.8473	0.8169	0.7114	0.7252	0.7959	0.8588	0.8626
DAGMM	**0.8738**	0.8011	0.7476	0.8249	0.7798	0.861	0.8647	0.8669	**0.9439**	0.9056	0.9172	0.871	0.8588	0.9439	0.9165	**0.9439**
USAD	**0.8539**	0.7834	0.8394	0.809	0.8267	0.8535	0.8313	0.8535	**0.9355**	0.8952	0.8043	0.7653	0.7928	0.9166	0.9289	0.9337
GDN	**0.8037**	0.763	0.792	0.7572	0.7548	0.7969	0.7742	0.7969	**0.9525**	0.8764	0.823	0.8601	0.9164	0.9488	0.9335	0.9488
TranAD	0.9714	0.9472	0.9643	0.9575	0.8733	0.9679	0.9485	**0.9717**	**0.9323**	0.8528	0.9069	0.9001	0.915	0.927	0.9309	0.9313
DOMI	**0.8849**	0.7914	0.7529	0.7731	0.7482	0.7608	0.8752	0.7608	**0.9316**	0.8247	0.8258	0.7953	0.8683	0.9241	0.9286	0.9241
SLAVAE	**0.8417**	0.7914	0.6368	0.7625	0.7196	0.7039	0.8158	0.7039	**0.8589**	0.8264	0.8011	0.6939	0.7136	0.7852	0.8427	0.7852
MTAD-GAT	**0.9414**	0.9109	0.8829	0.8983	0.7255	0.9407	0.9365	0.9407	**0.9757**	0.9265	0.9331	0.9238	0.6227	0.9714	0.9683	0.9715

The bold values indicate the maximum value in a row in a dataset.

OmniTransfer to create seven variants. C1: Only one base model is trained for transfer learning, and the data used to train the base model are randomly selected. C2: All metrics have the same weights when aligning phase shift and clustering. C3: Do not align the phase shift. C4: The base model is directly used for anomaly detection. C5: Use the full parameter transfer strategy for all MTS. C6: Use the partial parameter transfer strategy for all MTS. C7: Use the weighted Euclidean distance to select the transfer strategy. Table 7 shows the results of each variant.

Effect of Clustering. With an F_1 of lower than 0.57, the performance of C1 is far from satisfactory. The large difference between the base MTS and the target MTS makes transfer learning challenging. Clustering can effectively group MTS with similar shapes, making it easy to transfer the knowledge of base MTS to target MTS.

Effect of Metric Weighting. C2 has relatively poor performance on both datasets regardless of the algorithms. The reason is that aperiodic metrics are irregular and can have a negative impact on clustering. Generally, the distance between two aperiodic metrics can be considerable even though the periodic metrics in the same entities are relatively similar. Besides, aperiodic metrics can make the target MTS and the corresponding cluster centroid not being very similar. Therefore, it is indispensable to weighting these aperiodic metrics.

Effect of Phase Alignment. C3 needs more training overhead and has a poor performance than *OmniTransfer*. Without phase alignment, the diversity of MTS patterns increases, resulting in more clusters and more base models. Therefore, the training overhead increases dramatically. Additionally, it is difficult to match the target data with the appropriate cluster without phase alignment. Transfer learning cannot be effective when the target data and the base model training data differ significantly.

Effect of Transfer Learning. C4 directly uses the base model of each cluster for anomaly detection. Although the target MTS should be reasonably similar to its matching cluster centroid, there are still many tiny differences. These differences make the F_1 relatively poor. It is indispensable to transfer model parameters and fine-tune the base model.

Effect of Adaptive Transfer Strategy. OmniTransfer with an adaptive transfer strategy performs better than using a fixed transfer strategy. When the target MTS and its corresponding base cluster centroid are similar, it is better to transfer full parameters because more parameters can carry more valuable knowledge learned from the offline training stage. However, many target MTS have relatively large shape differences compared to the centroid. It is better to transfer partial parameters to avoid negative transfer problems. By automatically selecting the best transfer strategy for each target MTS, *OmniTransfer* gets the highest F_1.

Effects of the Distance Measurement of Adaptive Transfer Strategy. Compared with C7, *OmniTransfer* has an improvement in the detection performance on most models. The weighted Euclidean distance measures the difference between the target MTS and the cluster centroid. However, we aim to

Table 8. Comparison of Clustering Methods

Model	Dataset1					Dataset2				
	OmniTransfer	TICC	FCFW	M2PCA	SPCA+AED	*OmniTransfer*	TICC	FCFW	M2PCA	SPCA+AED
OmniAnomaly	**0.9721**	0.7209	0.7384	0.7341	0.6697	**0.974**	0.6339	0.6281	0.6494	0.6485
InterFusion	**0.9047**	0.6097	0.5528	0.6988	0.6949	**0.9235**	0.7006	0.6313	0.7897	0.8442
SDFVAE	**0.8512**	0.7231	0.7137	0.7399	0.71	**0.8673**	0.8327	0.8483	0.861	0.8663
DAGMM	**0.8738**	0.8537	0.8225	0.7886	0.8420	**0.9439**	0.8825	0.8965	0.8922	0.8937
USAD	**0.8539**	0.8167	0.8216	0.8157	0.8128	**0.9355**	0.9004	0.9014	0.8879	0.8933
GDN	**0.8037**	0.8022	0.7934	0.8033	0.7793	**0.9525**	0.8806	0.8778	0.8877	0.8599
TranAD	**0.9714**	0.9499	0.95	0.9564	0.9524	**0.9323**	0.8492	0.8426	0.8439	0.831
DOMI	**0.8849**	0.7439	0.7361	0.7515	0.7264	**0.9316**	0.8249	0.8628	0.8527	0.8362
SLAVAE	**0.8417**	0.7196	0.6709	0.7288	0.7047	**0.8589**	0.8251	0.8283	0.8477	0.8336
MTAD-GAT	**0.9414**	0.8933	0.9032	0.9009	0.9028	**0.9757**	0.9367	0.9355	0.9276	0.9296

The bold values indicate the maximum value in a row in a dataset.

transfer the knowledge in the base model to help detect anomalies in the target MTS. The $DiffScore$ measures the degree of match between the target MTS and the knowledge in the base model.

5.5 Effectiveness of the Clustering Method

To verify the advantages of the W-HAC in *OmniTransfer*, we select four baseline clustering methods for comparison: TICC [13], FCFW [20], Mc2PCA [19], and SPCA+AED [40]. We replace the clustering methods in *OmniTransfer* and use the anomaly detection performance as the clustering performance. Table 8 shows that the W-HAC's F_1 improves by 15.35% and 12.80% averagely on two datasets. We try to analyze the reasons. In general, these methods cannot resist noise and anomaly interference, and some cannot capture MTS shape features well. Specifically, TICC is only suitable for short-term data, and it is difficult for TICC to cluster 1-day data. FCFW uses all metrics data, which can be interfered with aperiodic metrics. SPCA+AED and Mc2PCA use PCA to reduce the dimension of MTS, which loses a lot of shape information, resulting in inaccurate clustering.

5.6 Effect of Transfer Strategy Selection Threshold

Recall that β is the threshold of $DiffScore$. To investigate the effect of β, we conduct experiments with different values of β. Figure 9 shows that the performance of *OmniTransfer* is higher than the worse single transfer strategy on two datasets, regardless of the value of β. Moreover, it can meet or even surpass the better single transfer strategy. The performance of *OmniTransfer* on OmniAnomaly, InterFusion, SDFVAE, DOMI, and SLAVAE is sensitive to β, while other models are insensitive. For insensitive models, the value of β will not greatly impact the experimental results. Therefore, we can easily obtain the β that makes each model perform well. For sensitive models, we randomly select some entities (e.g., 20 segments with 1 day of 200 entities) in the dataset to get β', which can reach the optimal β performance. Short-term segments also allow us to determine β' earlier. We invited three experienced operators, and it takes about 1 day to label 20 entities' data, so we only need less than 3 days of manpower to start the model, compared to 30 days for labeling 200 entities saves a lot of labor costs.

6 Related Work

6.1 MTS Clustering

There have been many studies on MTS clustering. SPCA+AED [40] proposes a hybrid method based on the PCA similarity factor (SPCA) and the **average-based Euclidean distance (AED)**.

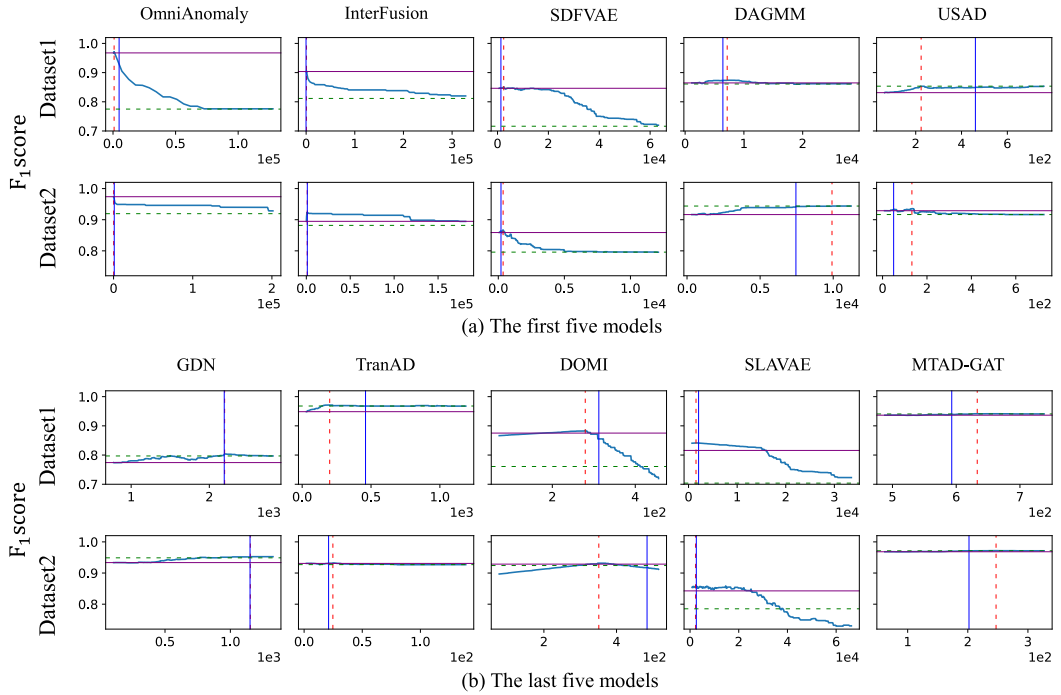

Fig. 9. The performance of different β. (The horizontal axis represents the value of β. The red vertical dotted line denotes the optimal β, the blue vertical solid line denotes the β' determined by some segments, the green horizontal dotted line denotes the performance of full parameter transfer strategy, the horizontal solid line denotes the performance of partial parameter transfer strategy).

Nevertheless, employing SPCA results in the loss of a significant amount of crucial information, and AED cannot address the phase shift problem. **Toeplitz inverse covariance-based clustering (TICC)** [13] focuses on the subsequences segmentation and clustering of MTS simultaneously. Segmentation is unnecessary in anomaly detection, and it is challenging for TICC to deal MTS with more than 100 time points (about 1 day). Mc2PCA [19] constructs common projection axes as the prototype of each cluster and uses the reconstruction error to assign the MTS. This method only considers the similarity within clusters, without the dissimilarity among clusters. FCFW [20] uses a fuzzy c-means method based on feature-weighted distance combining DTW and SBD. The time complexity of DTW is too high, which is unacceptable for large-scale software systems. Moreover, DTW and SBD consider each metric's shape features, which can be interfered with aperiodic metrics. CTF [35] uses the low-dimensional features extracted by the pre-trained anomaly detection model, which is task-specific and model structure-specific model [33]. OmniCluster [45] compresses the temporal dimension of MTS with a 1D convolutional **auto-encoder (AE)** and uses a three-step feature selection strategy to remove aperiodic metrics. However, the compressing and feature selection stages lose a lot of useful information. And the feature selection depends on an empirical threshold, which is not general.

6.2 MTS Anomaly Detection

There have been many studies on MTS anomaly detection. Both USAD and TranAD adversely train AE, and they take advantage of the stability of AE and the ability to isolate anomalies of

GAN. DAGMM combines AE and **Gaussian mixture model (GMM)**. It uses an AE to generate the low-dimensional features and reconstruction errors and feeds them into GMM to get the anomaly score. TranAD uses a sequence encoder with self-attention to shorten the inference time. OmniAnomaly uses the RNN+VAE structure to model the temporal dependence and stochasticity in MTS. Both SDFVAE and InterFusion adopt the structure of RNN+CNN+VAE. SDFVAE resists noise by modeling time-invariant and time-varying features. InterFusion employs a two-view embedding and prefiltering strategy to explicitly learn the inter-metric and temporal dependencies. DOMI uses VAE+GMM to model the intrinsic multimodality of data by obtaining complex latent representations. SLAVAE uses semi-supervised VAE and active learning to enhance robustness. GDN and MTAD-GAT are both prediction-based models. GDN uses structure learning and GNN to model the correlation between metrics. MTAD-GAT leverages two parallel graph attention layers to learn the relationships between different metrics dynamically.

However, the above models face high training overhead when dealing with large-scale MTS data and long initialization time. CTF, OmniCluster, JumpStarter, and Uni-AD successfully reduce the training overhead. CTF provides a solution to reduce training overhead for RNN+VAE models [33], but it is not universal to other models. OmniCluster is a model-agnostic framework that can reduce the training overhead. It trains a model for each cluster and directly uses it for anomaly detection. However, it performs poorly when the shape of the target MTS and the cluster centroid differs. JumpStarter uses the *Compressed Sensing* to reduce the model initialization time. However, due to only using short-term data and a simple model structure, it cannot capture complex patterns and long temporal dependence. Uni-AD uses a model-sharing mechanism and transformer layers to model large-scale time series. However, it does not work well when different entities' patterns diverge. In short, none of the above solutions can reduce the training cost and model initialization time while improving most SOTA models' detection results.

7 Discussion

In developing *OmniTransfer*, we have learned the following lessons. (1) The strength of periodicity is very important for MTS clustering. The information obtained from weak periodicity metrics is limited and can even seriously affect clustering. (2) The idea of adaptive transfer strategy and novel distance measurement for transfer strategy selection can ensure that we can achieve the optimal transfer strategy for each target MTS. (3) Reducing the number of detection models, reducing the scale of training data and accelerating model convergence speed are all effective solutions to reduce training overhead.

In addition, we have some ideas for future work. (1) We design a model-agnostic framework *OmniTransfer* for large-scale anomaly detection. The same ideas and key techniques can be used to reduce model initialization time and training overhead for other tasks, such as the prediction and classification of large-scale MTS. (2) The weights employed in the W-HAC method can be derived from prior knowledge or other methodologies, enhancing the clustering process by incorporating additional information and improving accuracy. (3) In practical applications, β can be randomly selected at first and be continuously updated with the supplement of data and manual feedback. The detection accuracy of the model could gradually increase.

There are also some limitations in our work. We directly only use full parameter transfer and partial parameter transfer strategies. When using partial parameter transfer strategies, the parameters of which layer to transfer are fixed for each model. It can be further investigated how to choose which part of the parameters to transfer or to transfer different parts of the parameters for different data to improve the effectiveness of transfer learning. Nevertheless, the adaptive strategy has achieved good performance for most models, and a simple and elegant method is better than complicated methods for a general framework.

8 Conclusion

This article first clearly points out the limitations of existing methods in large-scale MTS scenarios. And we propose *OmniTransfer*, a model-agnostic, unsupervised, and efficient anomaly detection framework to address these limitations. *OmniTransfer* uses transfer learning to reduce model initialization time and training overhead effectively. We propose W-HAC to reduce the interference of aperiodic metrics in clustering and improve the effect of transfer learning. Our experiment results using real-world datasets from a large web content service provider and a network operator show that *OmniTransfer* can reduce the initialization time by 46.49% and improve training efficiency by 74.51% compared to baseline models. We believe *OmniTransfer* is useful for large IT infrastructure, especially when monitoring millions of services that change frequently. *OmniTransfer* makes the anomaly detection models as rapidly deployable and cost-effective as possible for the large-scale and changing MTS.

References

[1] Minghan Liang, Zeyu Che, and Zhiyao Luo. 2024. Retrieved from https://anonymous.4open.science/r/OmniTransfer

[2] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A. Zuluaga. 2020. USAD: UnSupervised Anomaly Detection on Multivariate Time Series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. ACM, New York, NY, 3395–3404.

[3] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. 2016. Factors of Transferability for a Generic ConvNet Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38, 9 (2016), 1790–1802.

[4] Andrea Borghesi, Martin Molan, Michela Milano, and Andrea Bartolini. 2022. Anomaly Detection and Anticipation in High Performance Computing Systems. *IEEE Transactions on Parallel and Distributed Systems* 33, 4 (2022), 739–750. DOI: https://doi.org/10.1109/TPDS.2021.3082802

[5] Bin Cao, Sinno Jialin Pan, Yu Zhang, Dit-Yan Yeung, and Qiang Yang. 2010. Adaptive Transfer Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* 24, 1 (Jul. 2010), 407–412.

[6] Eva Cetinic, Tomislav Lipic, and Sonja Grgic. 2018. Fine-Tuning Convolutional Neural Networks for Fine Art Classification. *Expert Systems with Applications* 114 (2018), 107–118.

[7] Liang Dai, Tao Lin, Chang Liu, Bo Jiang, Yanwei Liu, Zhen Xu, and Zhi-Li Zhang. 2021. SDFVAE: Static and Dynamic Factorized VAE for Anomaly Detection of Multivariate CDN KPIs. In *Proceedings of the Web Conference 2021 (WWW '21)*. ACM, New York, NY, 3076–3086.

[8] Alain De Cheveigné and Hideki Kawahara. 2002. YIN, a Fundamental Frequency Estimator for Speech and Music. *The Journal of the Acoustical Society of America* 111, 4 (2002), 1917–1930.

[9] Ailin Deng and Bryan Hooi. 2021. Graph Neural Network-Based Anomaly Detection in Multivariate Time Series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35, 4027–4035.

[10] Li Dongwen, Zhang Shenglin, Sun Yongqian, Guo Yang, Che Zeyu, Chen Shiqi, Zhong Zhenyu, Liang Minghan, Shao Minyi, Li Mingjie, et al. 2023. An Empirical Analysis of Anomaly Detection Methods for Multivariate Time Series. In *2023 IEEE International Symposium on Software Reliability Engineering (ISSRE '23)*. IEEE Computer Society, Florence, Italy, 57–68.

[11] Yu Gan, Yanqi Zhang, Dailun Cheng, Ankitha Shetty, Priyal Rathi, Nayan Katarki, Ariana Bruno, Justin Hu, Brian Ritchken, Brendon Jackson, et al. 2019. An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud & Edge Systems. In *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '19)*. ACM, New York, NY, 3–18.

[12] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. 2019. SpotTune: Transfer Learning through Adaptive Fine-Tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR '19)*, 4800–4809.

[13] David Hallac, Sagar Vare, Stephen Boyd, and Jure Leskovec. 2017. Toeplitz Inverse Covariance-Based Clustering of Multivariate Time Series Data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 215–223.

[14] Zilong He, Pengfei Chen, and Tao Huang. 2022. Share or Not Share? Towards the Practicability of Deep Models for Unsupervised Anomaly Detection in Modern Online Systems. In *Proceedings of the 2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE '22)*, 25–35. DOI: https://doi.org/10.1109/ISSRE55969.2022.00014

[15] Tao Huang, Pengfei Chen, and Ruipeng Li. 2022. A Semi-Supervised VAE Based Active Anomaly Detection Framework in Multivariate Time Series for Online Systems. In *Proceedings of the ACM Web Conference 2022 (WWW '22)*. ACM, New York, NY, 1797–1806. DOI: https://doi.org/10.1145/3485447.3511984

[16] Mohammad Ali Humayun, Hayati Yassin, Junaid Shuja, Abdullah Alourani, and Pg Emeroylariffion Abas. 2022. A Transformer Fine-Tuning Strategy for Text Dialect Identification. *Neural Computing and Applications* 35 (2022), 1–10.

[17] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding. arXiv:1802.04431. Retrieved from https://doi.org/10.1145/3219819.3219845

[18] James M. Joyce. 2011. Kullback-Leibler Divergence. In *International Encyclopedia of Statistical Science*. Springer, 720–722.

[19] Hailin Li. 2019. Multivariate Time Series Clustering Based on Common Principal Component Analysis. *Neurocomputing* 349 (2019), 239–247.

[20] Hailin Li and Miao Wei. 2020. Fuzzy Clustering Based on Feature Weights for Multivariate Time Series. *Knowledge-Based Systems* 197 (2020), 105907.

[21] Ze Li, Qian Cheng, Ken Hsieh, Yingnong Dang, Peng Huang, Pankaj Singh, Xinsheng Yang, Qingwei Lin, Youjiang Wu, Sebastien Levy, et al. 2020. Gandalf: An Intelligent, End-To-End Analytics Service for Safe Deployment in Large-Scale Cloud Infrastructure. In *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI '20)*. USENIX Association, Santa Clara, CA, 389–402.

[22] Zhihan Li, Youjian Zhao, Yitong Geng, Zhanxiang Zhao, Hanzhang Wang, Wenxiao Chen, Huai Jiang, Amber Vaidya, Liangfei Su, and Dan Pei. 2022. Situation-Aware Multivariate Time Series Anomaly Detection through Active Learning and Contrast VAE-Based Models in Large Distributed Systems. *IEEE Journal on Selected Areas in Communications* 40, 9 (2022), 2746–2765. DOI: https://doi.org/10.1109/JSAC.2022.3191341

[23] Zhihan Li, Youjian Zhao, Jiaqi Han, Ya Su, Rui Jiao, Xidao Wen, and Dan Pei. 2021. Multivariate Time Series Anomaly Detection and Interpretation Using Hierarchical Inter-Metric and Temporal Embedding. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 3220–3230.

[24] Zhihan Li, Youjian Zhao, Rong Liu, and Dan Pei. 2018. Robust and Rapid Clustering of KPIs for Large-Scale Anomaly Detection. In *Proceedings of the 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS '18)*, 1–10.

[25] Dewei Liu, Chuan He, Xin Peng, Fan Lin, Chenxi Zhang, Shengfang Gong, Ziang Li, Jiayu Ou, and Zheshun Wu. 2021. MicroHECL: High-Efficient Root Cause Localization in Large-Scale Microservice Systems. In *Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '21)*, 338–347.

[26] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. 2022. Frozen pretrained transformers as universal computation engines. In *AAAI Conference on Artificial Intelligence*. Retrieved from https://api.semanticscholar.org/CorpusID:250301249

[27] Meng Ma, Jingmin Xu, Yuan Wang, Pengfei Chen, Zonghua Zhang, and Ping Wang. 2020. AutoMAP: Diagnose Your Microservice-Based Web Applications Automatically. In *Proceedings of the Web Conference 2020 (WWW '20)*. ACM, New York, NY, 246–258.

[28] Minghua Ma, Shenglin Zhang, Junjie Chen, Jim Xu, Haozhe Li, Yongliang Lin, Xiaohui Nie, Bo Zhou, Yong Wang, and Dan Pei. 2021. {Jump-Starting} Multivariate Time Series Anomaly Detection for Online Service Systems. In *Proceedings of the 2021 USENIX Annual Technical Conference (USENIX ATC '21)*, 413–426.

[29] Minghua Ma, Shenglin Zhang, Dan Pei, Xin Huang, and Hongwei Dai. 2018. Robust and Rapid Adaption for Concept Drift in Software System Anomaly Detection. In *Proceedings of the 2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE '18)*, 13–24.

[30] Aditya P. Mathur and Nils Ole Tippenhauer. 2016. SWaT: A Water Treatment Testbed for Research and Training on ICS Security. In *Proceedings of the 2016 International Workshop on Cyber-Physical Systems for Smart Water Networks (CySWater '16)*, 31–36.

[31] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (2010), 1345–1359.

[32] Ravi K. Samala, Heang-Ping Chan, Lubomir Hadjiiski, Mark A. Helvie, Caleb D. Richter, and Kenny H. Cha. 2019. Breast Cancer Diagnosis in Digital Breast Tomosynthesis: Effects of Training Sample Size on Multi-Stage Transfer Learning Using Deep Neural Nets. *IEEE Transactions on Medical Imaging* 38, 3 (2019), 686–696.

[33] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. 2019. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2828–2837.

[34] Ya Su, Youjian Zhao, Ming Sun, Shenglin Zhang, Xidao Wen, Yongsu Zhang, Xian Liu, Xiaozhou Liu, Junliang Tang, Wenfei Wu, et al. 2022. Detecting Outlier Machine Instances through Gaussian Mixture Variational Autoencoder with One Dimensional CNN. *IEEE Transactions on Computers* 71, 4 (2022), 892–905.

[35] Ming Sun, Ya Su, Shenglin Zhang, Yuanpu Cao, Yuqing Liu, Dan Pei, Wenfei Wu, Yongsu Zhang, Xiaozhou Liu, Junliang Tang, et al. 2021. CTF: Anomaly Detection in High-Dimensional Time Series with Coarse-to-Fine Model Transfer. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 1–10.

[36] Nima Tajbakhsh, Jae Y. Shin, Suryakanth R. Gurudu, R. Todd Hurst, Christopher B. Kendall, and Michael B. Gotway. 2016. Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? *IEEE Transactions on Medical Imaging* 35, 5 (2016), 1299–1312.

[37] Lisa Torrey and Jude Shavlik. 2010. Transfer Learning. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. IGI Global, 242–264.

[38] Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. 2022. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data. arXiv:2201.07284. Retrieved from https://doi.org/10.48550/arXiv.2201.07284

[39] Grega Vrbančič and Vili Podgorelec. 2020. Transfer Learning with Adaptive Fine-Tuning. *IEEE Access* 8 (2020), 196197–196211.

[40] J. Wu, S. K. Nguang, J. Shen, G. Liu, and Y. G. Li. 2010. Robust H∞ Tracking Control of Boiler–Turbine Systems. *ISA Transactions* 49, 3 (2010), 369–375.

[41] Fanghua Ye, Zhiwei Lin, Chuan Chen, Zibin Zheng, and Hong Huang. 2021. Outlier-Resilient Web Service QoS Prediction. In *Proceedings of the Web Conference 2021 (WWW '21)*. ACM, New York, NY, 3099–3110.

[42] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How Transferable Are Features in Deep Neural Networks? In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (Eds.), Vol. 27, Curran Associates, Inc.

[43] Guangba Yu, Pengfei Chen, Hongyang Chen, Zijie Guan, Zicheng Huang, Linxiao Jing, Tianjun Weng, Xinmeng Sun, and Xiaoyun Li. 2021. MicroRank: End-to-End Latency Issue Localization with Extended Spectrum Analysis in Microservice Environments. In *Proceedings of the Web Conference 2021 (WWW '21)*. ACM, New York, NY, 3087–3098.

[44] Guangba Yu, Pengfei Chen, and Zibin Zheng. 2019. Microscaler: Automatic Scaling for Microservices with an Online Learning Approach. In *Proceedings of the 2019 IEEE International Conference on Web Services (ICWS '19)*, 68–75.

[45] Shenglin Zhang, Dongwen Li, Zhenyu Zhong, Jun Zhu, Minghan Liang, Jiexi Luo, Yongqian Sun, Ya Su, Sibo Xia, Zhongyou Hu, et al. 2022. Robust System Instance Clustering for Large-Scale Web Services. In *Proceedings of the ACM Web Conference 2022*, 1785–1796.

[46] Shenglin Zhang, Zhenyu Zhong, Dongwen Li, Qiliang Fan, Yongqian Sun, Man Zhu, Yuzhi Zhang, Dan Pei, Jiyan Sun, Yinlong Liu, et al. 2022. Efficient KPI Anomaly Detection through Transfer Learning for Large-Scale Web Services. *IEEE Journal on Selected Areas in Communications* 40, 8 (2022), 2440–2455.

[47] Xu Zhang, Qingwei Lin, Yong Xu, Si Qin, Hongyu Zhang, Bo Qiao, Yingnong Dang, Xinsheng Yang, Qian Cheng, Murali Chintalapati, et al. 2019. Cross-Dataset Time Series Anomaly Detection for Cloud Systems. In *Proceedings of the 2019 USENIX Annual Technical Conference (USENIX ATC '19)*, 1063–1076.

[48] Yongle Zhang, Junwen Yang, Zhuqi Jin, Utsav Sethi, Kirk Rodrigues, Shan Lu, and Ding Yuan. 2021. Understanding and Detecting Software Upgrade Failures in Distributed Systems. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles (SOSP '21)*. ACM, New York, NY, 116–131.

[49] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang. 2020. Multivariate Time-Series Anomaly Detection via Graph Attention Network. In *Proceedings of the 2020 IEEE International Conference on Data Mining (ICDM '20)*. IEEE Computer Society, Los Alamitos, CA, 841–850.

[50] Nengwen Zhao, Junjie Chen, Zhaoyang Yu, Honglin Wang, Jiesong Li, Bin Qiu, Hongyu Xu, Wenchi Zhang, Kaixin Sui, and Dan Pei. 2021. Identifying Bad Software Changes via Multimodal Anomaly Detection for Online Service Systems. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '21)*. ACM, New York, NY, 527–539.

[51] Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. 2023. One Fits All: Power General Time Series Analysis by Pretrained LM. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36, Curran Associates, Inc., 43322–43355. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2023/file/86c17de05579cde52025f9984e6e2ebb-Paper-Conference.pdf

[52] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2021. A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE* 109, 1 (2021), 43–76.

[53] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. 2018. Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection. In *International Conference on Learning Representations*.

Received 22 February 2024; revised 30 July 2024; accepted 2 October 2024