Pre-trained KPI Anomaly Detection Model Through Disentangled Transformer

Zhaoyang Yu
yu-zy20@mails.tsinghua.edu.cn
Tsinghua University & BNRist
Beijing, China

Changhua Pei*
chpei@cnic.cn
Computer Network Information Center, Chinese Academy of Sciences
Beijing, China

Xin Wang
x.wang@stonybrook.edu
Stony Brook University
New York, USA

Minghua Ma
minghuama@microsoft.com
Microsoft
Redmond, USA

Chetan Bansal
chetanb@microsoft.com
Microsoft
Redmond, USA

Saravan Rajmohan
saravar@microsoft.com
Microsoft
Redmond, USA

Qingwei Lin
qlin@microsoft.com
Microsoft
Beijing, China

Dongmei Zhang
dongmeiz@microsoft.com
Microsoft
Beijing, China

Xidao Wen
wenxidao@bizseer.com
BizSeer Technology
Beijing, China

Jianhui Li
lijh@cnic.cn
Computer Network Information Center, Chinese Academy of Sciences
Beijing, China

Gaogang Xie
xie@cnic.cn
Computer Network Information Center, Chinese Academy of Sciences
Beijing, China

Dan Pei
peidan@tsinghua.edu.cn
Tsinghua University & BNRist
Beijing, China

Abstract

In large-scale online service systems, numerous Key Performance Indicators (KPIs), such as service response time and error rate, are gathered in a time-series format. KPI Anomaly Detection (KAD) is a critical data mining problem due to its widespread applications in real-world scenarios. However, KAD faces the challenges of dealing with KPI heterogeneity and noisy data. We propose ***KAD-Disformer***, a **K**PI **A**nomaly **D**etection approach through **Dis**entangled Trans**former**. *KAD-Disformer* pre-trains a model on existing accessible KPIs, and the pre-trained model can be effectively "fine-tuned" to unseen KPI using only a handful of samples from the unseen KPI. We propose a series of innovative designs, including disentangled projection for transformer, unsupervised few-shot fine-tuning (*uTune*), and denoising modules, each of which significantly contributes to the overall performance. Our extensive experiments demonstrate that *KAD-Disformer* surpasses the state-of-the-art universal anomaly detection model by 13% in F1-score and achieves comparable performance using only 1/8 of the fine-tuning samples saving about 25 hours. *KAD-Disformer* has been successfully deployed in the real-world cloud system serving millions of users, attesting to its feasibility and robustness. Our code is available at https://github.com/NetManAIOps/KAD-Disformer.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**; **Machine learning**.

Keywords

Time-Series, Anomaly Detection, Disentangled Transformer

ACM Reference Format:
Zhaoyang Yu, Changhua Pei, Xin Wang, Minghua Ma, Chetan Bansal, Saravan Rajmohan, Qingwei Lin, Dongmei Zhang, Xidao Wen, Jianhui Li, Gaogang Xie, and Dan Pei. 2024. Pre-trained KPI Anomaly Detection Model Through Disentangled Transformer . In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671522

1 Introduction

Online service systems are playing essential and growing roles in our daily life. Some example systems are social networks, online shopping, mobile payment, and search engine. To guarantee the high quality and non-interrupted services, businesses are gradually relying on Key Performance Indicator (KPI) of time series data to pinpoint and tackle anomalies and other concerns [5, 36, 42]. Anomaly detection (AD) based on time series focuses on rapidly identifying and addressing irregularities, making it a hot topic within data mining communities [18, 28, 33–35, 43]. Traditional

*Changhua Pei is the corresponding author.

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671522

KPI anomaly detection methods are mostly rule-based and consist of massive handicraft thresholds. Although this type of anomaly detection is computationally efficient, the detection performance is far from satisfactory [19]. Besides, huge human efforts for deciding the considerable parameters of the rules make these methods unfeasible for large-scale online systems.

In recent years, many efforts have been devoted to model-based time series anomaly detection algorithms, including supervised approaches (e.g., Opprentice [19]), semi-supervised approaches (e.g., ACVAE [17]) and unsupervised approaches (e.g., Donut [33], DAEMON [4], MAD-SGCN [25], TranAD [29], AnomalyTrans [34]). The powerful machine learning techniques help them achieve superior performance compared with the classic methods such as Holt-Winters [2] and ARIMA [22].

However, real-world time series data are complex and exhibit significant variations across different domains. Consequently, existing anomaly detection algorithms employ specialized models tailored for **each** individual KPI. However, for large online systems with hundreds of thousands of KPIs, training individual models leads to prohibitive overhead. Furthermore, the long initialization time [20][1] of existing algorithms, determined chiefly by the amount of data used for fine-tuning the model to achieve satisfactory accuracy, render them impractical for rapidly changing services. We identify two criteria for practical anomaly detection algorithms to monitor large-scale online systems:

- **Generalizable Pre-Training**: The ability to detect anomalies across diverse KPI datasets using a universal model, which is pre-trained on the accessible datasets and can adapt to previously unseen datasets.
- **Unsupervised Few-Shot Fine-Tuning**: The ability to achieve a strong performance even with limited data after fine-tuning, minimizing the initialization time. This allows models to rapidly adapt to unseen KPIs or services.

However a naive pre-trained KPI anomaly detection model may suffer from obvious performance degradation with the large drift between the "KPIs for pre-training" and the "KPIs for fine-tuning". The KPIs for pre-training represent the accessible KPIs utilized to pre-train the model. The KPIs for fine-tuning refer to the KPIs that we want the model to detect anomalies for, but they are not known or accessed during the pre-training process. To solve the performance degradation problem, some existing anomaly detection models [21, 40, 41] adopt a two-stage approach: classifying KPIs into different groups, then fine-tuning models for each group. However, the grouping stage introduces computation overhead and the performance can be compromised if a KPI is inappropriately clustered. Here we need a universal pre-trained model with the power to effectively and quickly adapt to incoming KPIs without explicitly clustering. Yet facing real-world complex KPI data, we encounter three challenges:

- **High KPI diversity**: KPIs from thousands of applications and systems have diverse, non-stationary patterns. Pre-training a single model for such heterogeneous data is pretty challenging.
- **Tailored model adaptation**: There is a challenge to ensure the model to quickly adapt to incoming KPIs while maintaining the good performance for historical KPIs. A general model with limited capacity may be able to quickly adapt to incoming KPIs but at the cost of performance degradation on historical KPIs. On the other hand, a complex model may be more capable on historical KPIs but is not flexible to incoming KPIs.
- **Robustness to noisy data**: Rapidly adapting to incoming KPIs during fine-tuning demands high-quality data, yet KPI time series often contain noises. Efficiently achieving satisfactory performance on limited and noisy data of the KPI for fine-tuning poses a considerable challenge.

In this paper, we propose ***KAD-Disformer***, a **K**PI **A**nomaly **D**etection approach through **Dis**entangled Trans**former**. Different from the previous transformer-based KAD models [29, 34], we disentangle the projection matrices (W) of *query, key, value* in Transformer into W_{common} and $W_{personal}$. W_{common} focuses on the common projection patterns across different types of KPIs. $W_{personal}$ tries to learn the personalized projection patterns for individual KPIs through a very limited number of samples. To achieve this, we design the uTune mechanism for unsupervised few-shot fine-tuning. Assisted by the tailored two-stage gradient update mechanism of uTune, the personalized projection matrices are capable of rapidly adapting to the KPIs, while effectively preventing over-fitting during the fine-tuning process. Our model improves the F1-score of anomaly detection by considerable margins. The detailed evaluation also confirms that our model can quickly achieve a high F1-score with only 1/8 fine-tuning samples compared with the other existing methods.

Our contributions are summarized as follows:

- To the best of our knowledge, *KAD-Disformer* we proposed is the first pre-trained time series-based KPI anomaly detection model. Through careful selection of the model structure and optimization techniques, we significantly enhance the effectiveness and efficiency (*i.e.*, initialization time) of the anomaly detection algorithm.
- In *KAD-Disformer*, we disentangle the projection matrices in Transformer into common projection and personalized projection, respectively, to effectively trade-off between maintaining the model capacity and quick adaption to an incoming KPI simultaneously. The personalized projection is updated in our uTune mechanism to quickly fit to fine-tuning samples with little risk of over-fitting.
- In *KAD-Disformer*, we design the adapter layers and denoising reconstruction mechanism to improve the detection accuracy.
- We conduct comprehensive evaluation not only to show the overall performance but also to measure the contributions of each part of *KAD-Disformer* for overall performance (Section 5).
- We have deployed *KAD-Disformer* to a large-scale real-world online service system, helping them maintain high-quality service for months. The codes of this paper is released at https://github.com/NetManAIOps/KAD-Disformer.

2 Related Work

Recent advancements in KPI anomaly detection have led to the development of numerous methodologies, broadly classified into

[1]The initialization time indicates the time lag between when operators collect fine-tuning data for the anomaly detection model and when it achieves satisfactory accuracy for online detection, mainly decided by the number of data points used.

supervised, unsupervised, and semi-supervised approaches [6, 11, 17, 18, 28, 33, 34, 40, 41].

Supervised methods, like Opprentice [19], leverage KPI patterns and labels akin to binary classifiers, incorporating techniques such as random forests [1]. Despite their effectiveness in various fields, the scarcity of high-quality labels limits their applicability in real-world KPI anomaly detection [12].

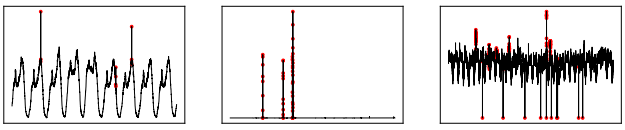
Unsupervised methods, evolving from statistical models like ARIMA [22] and Holt-Winters [2], to more sophisticated deep-learning-based approaches [3, 7, 11, 18, 28, 29, 33, 34, 38], have shown significant improvements. These methods, especially those utilizing Transformer architectures [29, 34], excel by learning from normal patterns to identify anomalies.

Though deep-learning-based unsupervised methods' performance is superior to the traditional methods, the overhead brought by re-training on large-scale different KPIs makes them infeasible in large-scale online service systems [27]. Therefore, several transferable methods have been proposed these days. These methods can learn from existing large-scale KPI data and efficiently transfer the model to fit an incoming KPI [40, 41]. For example, ATAD [41], a cluster-based semi-supervised method, extracts the features from KPIs with the hand-crafted rules and groups the historical KPIs. When a new KPI comes, ATAD will assign this KPI to a group and then ask for partial labels to fine-tune the classifier. However, the performance of ATAD is unstable and highly related to the clustering algorithm and the quality of labels. All these transferable methods rely on extra information such as clustering to guarantee the detection performance.

3 Preliminaries

3.1 KPI Anomaly Detection

Since time series data in online service systems are largely affected by service schedules or user behaviors, most of them have shown the property of seasonality [15]. Time series collected from real-world production environments inevitably have noises. Therefore, the normal patterns of seasonal time series have two parts: 1) normal seasonal patterns with local variations, and 2) noises with some kind of distribution [40]. For univariate time series, people usually regard the data points that do not follow the normal patterns as anomaly points [40] (e.g., spikes or dips).



(a) seasonal (b) stable (c) unstable

Figure 1: Examples of time series from a global Internet company. The red points mark the anomalies.

For the sake of brevity, we use the notation similar to [14]. Given a time series data $X = [x_0, x_1, x_2, \cdots, x_t]$ and label series $Y = [y_0, y_1, y_2, \cdots, y_t]$, where $x_i \in \mathbb{R}, y_i \in \{0, 1\}, t \in \mathbb{N}$. X denotes the whole time series data array, and x_i denotes the metric value at time i. Y denotes the labels of time series X. $y_i = 0$ indicates that data point at i is normal and $y_i = 1$ indicates the data point at i is an anomaly. Besides, $X_{i,j} = [x_i, x_{i+1}, \cdots, x_{j-1}, x_j]$ indicates a window of X from time i to j.

With the above notations, we define the time series anomaly detection as follows:

Given a time series $X = [x_0, x_1, x_2, \cdots, x_t]$, the goal of the KPI anomaly detection is to predict the corresponding label series $Y = [y_0, y_1, y_2, \cdots, y_t]$. When predicting y_i, only $X_{0,i} = [x_0, \cdots, x_{i-1}, x_i]$ can be used, since the sub-sequence after time i is unknown at time i.

3.2 Few-Shot in KAD

In the KAD domain, there is currently no clear definition of "few-shot". In real-world environments, KPI data is continuously generated in chronological order. Therefore, when integrating new KPIs and deploying KAD models, there is an initialization time. The time waiting for new KPI data to be generated (*i.e.*, collecting data for fine-tuning) occupies the majority of this initialization time. This period usually spans several dozen or even hundreds of hours, because this time is directly aligned with real-world time, and the newly integrated KPIs do not have historical data [37].

Based on the above observation, we propose that the few-shot capability of KAD can be measured from two perspectives. First, whether KAD can achieve better anomaly detection performance with an equal amount of fine-tuning data. Second, whether less fine-tuning data is needed to achieve competitive anomaly detection performance. Both perspectives imply that less fine-tuning data can be collected to achieve satisfactory KAD performance, allowing for faster model deployment and improved efficiency.

4 Methodology

In this section, we first give a brief overview of *KAD-Disformer* from the aspects of architecture and workflow. Then we give detailed introductions to the modules of Disentangled Projection Matrices (denoted as DPM), uTune, and Denoising Reconstruction respectively. The design of Transformer is introduced together with DPM and the design of various adapters such as Series Adapter and Encoder Adapter is introduced together with uTune.

4.1 *KAD-Disformer* Model Overview

KAD-Disformer follows an encoder-decoder architecture and the overall architecture is shown in Figure 2. Transformer [31] can effectively capture the time dependency like RNN-based models but is more parallelizable. Transformer-based models have excellent generalizability, and a well-trained model can be effectively transferred to many other datasets and tasks through fine-tuning [24]. Inspired by these advantages, we exploit the Transformer framework as a base to tackle universal anomaly detection tasks.

The encoder-decoder architecture is widely used in KPI anomaly detection area [2, 3, 18, 28, 33, 34, 40]. A popular hypothesis of unsupervised encoder-decoder models for anomaly detection was proposed in [33], *i.e.*, given training data, the model can learn normal patterns through dimensionality reduction. No label is needed in this process, and all the knowledge is learned from the raw data automatically.

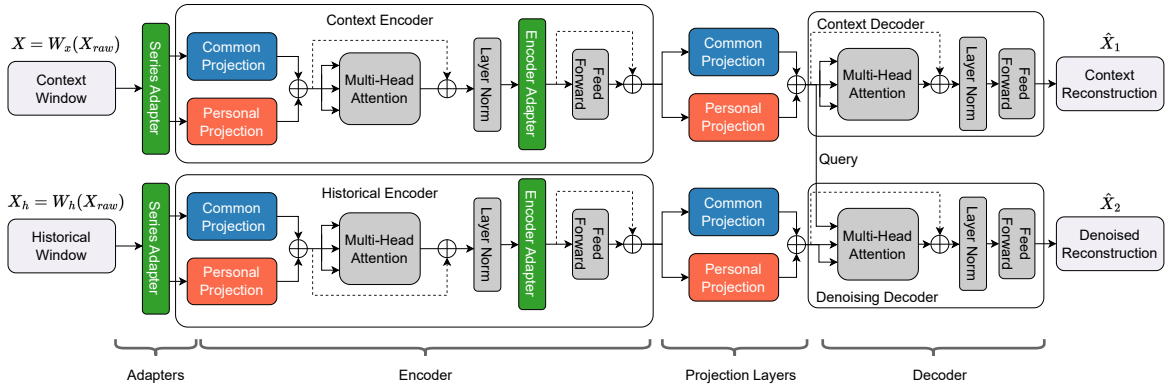


Figure 2: The overall architecture of *KAD-Disformer*.

4.2 Workflow of *KAD-Disformer*

Initially, we need to perform a data preprocessing on the raw KPI data for our model, *KAD-Disformer*. In contrast to existing methods [3, 11, 18, 33], our approach involves two sliding windows for the raw KPI data embedding. For each KPI input vector $X_{raw} = [x_0, x_1, x_2, \cdots, x_N]$, we apply two sliding windows to convert the input vector to a matrix. W_x acts as a context window with a stride of 1, with its purpose being the aggregation of local data information. On the other hand, W_h serves as a historical window, and its stride often bears some form of intrinsic significance, such as the KPI's period (denoted as p). The stride's value for this historical window can be decided by the users, based on their domain-specific knowledge, with the default stride being the KPI period as calculated by the Fast Fourier Transform (FFT) [23]. The primary goal of this historical window is to effectively recognize and retain the long-term dependencies inherent in the time series data.

The following workflow of *KAD-Disformer* consists of pre-training, fine-tuning, and inference. During the pre-training phase, we randomly initialize the whole parameters of *KAD-Disformer*, and each iteration updates all the parameters in the model. The detailed procedure is indicated in Appendix B.

As for fine-tuning, we design a uTune mechanism. The detail of uTune is in Section 4.4. uTune mechanism follows the adapter-based fine-tuning paradigm [10] and only updates the partial parameters of *KAD-Disformer*.

4.3 Disentangled Projection Matrices

The core component of the Transformer architecture is the Attention mechanism. Specifically, the original multi-head attention approach, as described in [31], employs three learnable linear projections W^Q, W^K, W^V to map the raw data X to Q (query), K (key), and V (value) matrices into distinct higher-dimensional spaces. This transformation is outlined in Equation (1).

$$Q = XW^Q, K = XW^K, V = XW^V \tag{1}$$

In the context of *KAD-Disformer*, we propose a novel approach aimed at decoupling the training of the base model from that of the task-specific model. To achieve this, we disentangle the linear projection W into two separate components: a generalization projection, denoted W_{common}, and a personalized projection, $W_{personal}$. W_{common} is only updated in the pre-training phase, delivering the acquired knowledge from the pre-training stage to the fine-tuning stage. Conversely, $W_{personal}$ is iteratively updated during both the pre-training and fine-tuning phases. The primary function of $W_{personal}$ is to assimilate knowledge specific to a particular KPI, thereby enhancing the "learning to learn" performance on the fine-tuning data. We delve into further detail regarding the updating of $W_{personal}$ in Section 5.5. Our method, referred to as disentangled dot-product attention, is formalized in Equation (2).

$$\begin{aligned} \text{Attention}(Q, K, V) &= \text{softmax}(\frac{QK^T}{\sqrt{w}})V \\ \text{Where } Q &= X(W^Q_{common} + W^Q_{personal}) \\ K &= X(W^K_{common} + W^K_{personal}) \\ V &= X(W^V_{common} + W^V_{personal}) \end{aligned} \tag{2}$$

where w is the size of our sliding window.

Why disentangle? The main advantage of disentangling projection matrices in universal AD is alleviating the overfitting problem in the fine-tuning procedure and keeping the knowledge learned from the pre-trained data for fine-tuning. If all the projection matrices are updated with fine-tuning data without disentangling, the projection matrices are easily overfitted. However, the small-scale data usually can not reflect all the patterns of the incoming KPIs. In disentangled projection matrices, W_{common} still stores the knowledge from the pre-training data after fine-tuning, so the attention is not easily overturned from the original projection matrices. Besides, our uTune can also alleviate the overfitting problem, which is discussed in Section 4.4.

Why projection matrices? The primary rationale behind our design choice to disentangle projection matrices lies in the fact that projection matrices constitute the core structure within a Transformer. Hence, these matrices are highly information-dense and important within a Transformer. W_{common} primarily takes responsibility for employing the knowledge obtained from pre-training to compute attention, whereas $W_{personal}$'s role is to make fine adjustments based on the characteristics of the current KPI. This approach thus achieves a balance between retaining pre-trained knowledge and utilizing the intrinsic feature of the KPI.

4.4 uTune

The model adaptation for unseen incoming KPI data is a great challenge in the universal AD area. To solve this problem, we propose a two-stage adapter-based fine-tuning mechanism called **uTune**. To better illustrate the functioning of uTune, we center our discussion around two key questions: "What to update?" and "How to update?".

4.4.1 What to update? $W_{personal}$ and adapters are the components updated during the uTune. The function of W_{person} has been thoroughly elucidated in Section 4.3. Our focus now shifts to the adapters within *KAD-Disformer*. The main purpose of the Adapter module is to maximize the utilization of the model's parameters from pre-training phase when handling unseen KPI data. As illustrated in Figure 2, there are two adapters: **Series Adapter** and **Encoder Adapter**.

Series Adapter layers take the raw fine-tuning data as input to learn a time series data adapter, making the model fit the incoming new time series data. To better adapt to the data and improve the anomaly detection performance, we incorporate a time series decomposition module [32] into the Series Adapter to eliminate the noises and produce more clean time series data. We decompose the time series into two parts: the seasonal part and the trend part. We apply the average polling window to compute the seasonal part of the time series and keep the residual as the trend part shown in Equation (3).

$$X_{\text{seasonal}} = \text{AvgPool}(X), \quad X_{\text{trend}} = X - X_{\text{seasonal}} \tag{3}$$

Average pooling can efficiently be executed in the neural network with little overhead. And then, the downstream modules of the time series decomposition are two feed-forward networks fed by the seasonal part and trend part, respectively. The feed-forward networks provide the learnable parameters for fine-tuning the time series. After being transferred by the feed-forward networks, we add the seasonal and trend part together and send the output to downstream.

Encoder Adapters are located in the encoder module (Figure 2). These layers are simple and efficient fully connected layers designed to better perform deep-level adaptation when the Encoder has multiple layers. In practice, the Encoder is often a stack of multiple layers, and the Series Adapter only acts on the input stage. As the number of Encoder layers increases, the effectiveness of the Series Adapter diminishes, so we introduce the Encoder Adapter to facilitate the effective utilization of the common knowledge in the deeper layers.

4.4.2 How to update? Our design is inspired by the concept of First-Order Model-Agnostic Meta-Learning (FOMAML) [8], we do not directly apply it but rather optimized it according to the characteristics of our task. In each iteration, we sample two mini-batches of data of the same size. Batch x_1 is from the unseen KPI X prepared for fine-tuning, and batch x_2 is from the existing pre-training datasets D_S. Given a loss function L, *KAD-Disformer* makes a feed-forward with x_1 and evaluates the loss L_1 in the first stage. And then *KAD-Disformer* makes the backpropagation to update the parameters of $W_{personal}$ and the aforementioned adapter layers (Line 5 to Line 6 in Algorithm 2). The first stage focuses on the unseen KPI and pushes the pre-trained base model to learn the personal information from the unseen KPI. We call the first stage the personalization stage.

In the second stage, we use x_2 to make feed-forward and evaluate the loss L_2. At this time, we don't use L_2 to make the backpropagation but compute a final loss as Equation (4).

$$L = \alpha L_1 + (1 - \alpha) L_2 \tag{4}$$

where $\alpha \in [0, 1]$ is a hyper-parameter balancing the weights of performance on new data (L_1) and existing data (L_2). According to our experience, we set α to 0.5, and the sensitivity analysis of α is in Appendix C. Then, we use L to make backpropagation. The second stage focuses on the overall data's performance and prevents $W_{personal}$ and adapter layers from overfitting the unseen KPI, making the model learn from new and existing data simultaneously. We call the second stage generalization stage. The pseudo code of the uTune can be found in Appendix B. The two-stage update can effectively reduce the converging time and simultaneously achieve high performance.

The personalization stage makes *KAD-Disformer* fit the unseen KPI data fastly while guaranteeing performance. Recall that the fine-tuning procedure is easily overfitted due to the small scale of fine-tuning data, which can lead to performance degradation after deployment. The generalization stage utilizes the knowledge from both new data and existing data to make the backpropagation in order to alleviate the overfitting problem. Moreover, different KPI data may share some common knowledge, *i.e.*, W_{common}. Thus, even if the small scale of unseen KPI data can not reflect the full features, this shared common knowledge can help the model better understand the features of unseen KPIs.

The main difference between our method and FOMAML is that we do not update the initial parameters θ uniformly after calculating the gradients of all tasks. Instead, we directly update the parameters to obtain θ^* after computing L_1, and then update again based on θ^*. Our design aims to better and faster adapt to unseen KPIs. The original FOMAML typically deals with multiple tasks, and its concept is to obtain more common knowledge from multiple few-shot tasks to achieve few-shot learning. In our scenario, each fine-tuning is actually for a specific unseen KPI, *i.e.*, the number of tasks is 1. Therefore, our focus is on quickly adapting to the current KPI. We directly update θ using the gradient of L_1 to obtain θ^*, and then continue updating based on θ^* at the end to prevent overfitting to the small amount of unseen KPI data and incorporate historical data.

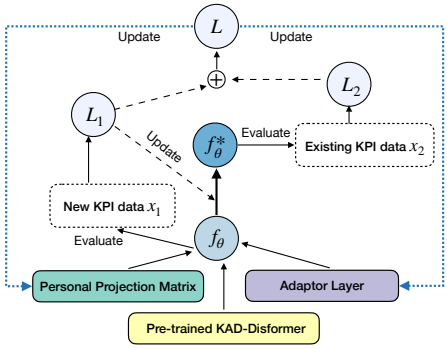


Figure 3: Demonstration of how the uTune works.

4.5 Denoising Reconstruction

The core idea of denoising reconstruction is to reconstruct a denoised time series rather than the original one. In the KAD task, a denoised reconstruction is more effective in detecting anomalies. The reason is that unsupervised methods use the error between the original data and the model's output to detect the anomaly [18, 28, 33, 34]. The main goal of the model is to learn the normal time series patterns from the original data. However, large-scale training data inevitably have some anomalies and noises in the real-world environment. Therefore, it is impossible for the model to learn from the absolute normal data. To tackle this problem, we design the denoising reconstruction mechanism to reconstruct a denoised time series to better distinguish the normal data and anomalies.

In the denoising reconstruction mechanism, there are two data flows. The first one is the context data flow, which captures the context information of the data point. The stride of the context sliding window is 1. The second one is the history data flow. The history data flow can capture the long-term dependency of the time series. The stride of the historical window can be the period of the time series or a value with physical meaning determined by the user. The goal of the historical window is to provide the denoising decoder with the historical information of the time series.

The input of the context decoder comes totally from the context encoder without historical information. We hope the output of the context decoder is as close as possible to the original time series. The input of the denoising decoder consists of two parts: one part is from the history encoder, and the other part is from the context encoder. The matrices K and V come from the history encoder, and the matrix Q comes from the context encoder. The goal is to use the same context query matrix Q to query the history information and make history knowledge help reconstruct a denoised KPI.

The loss of denoising reconstruction has two parts from the denoising decoder and context decoder, respectively. $\hat{X}_1$ denotes the output of the context decoder, and $\hat{X}_2$ denotes the output of the denoising decoder. The metric used to evaluate the reconstruction performance is Mean Squared Error (MSE), which is widely used in time series anomaly detection [11, 29, 34].

$$L = |L_{\text{context}}| + |L_{\text{denoised}}| \tag{5}$$

$$L_{\text{context}} = \text{MSE}(X, \hat{X}_1), \quad L_{\text{denoised}} = \text{MSE}(X, \hat{X}_2) \tag{6}$$

The final output for calculating anomaly scores is the average of $\hat{X}_1$ and $\hat{X}_2$.

5 Experiments

In this section, we evaluate *KAD-Disformer* using various KPI anomaly detection datasets collected from the real-world environment to answer the following questions.

- RQ1: What is the effectiveness of *KAD-Disformer*?
- RQ2: Can *KAD-Disformer* quickly achieve a desirable performance after being tuned with a small number of samples?
- RQ3: Is the design of disentangled projection helpful to improve the effectiveness of *KAD-Disformer*?
- RQ4: Is the uTune mechanism helpful to improve the effectiveness of *KAD-Disformer*?

As for the baseline models, we select one classic statistic method and five deep learning-based models, two of which employ their own tailored fine-tuning techniques. Firstly, the ARIMA model [22] is a conventional statistical anomaly detection approach that enjoys widespread industrial use. Within the deep learning spectrum, we choose LSTM-NDT (RNN-based) [11] and Donut (VAE-based) [33], both of which have achieved significant popularity and extensive application in various industrial scenarios. AnomalyTrans (Transformer-based) [34] represents the state-of-the-art unsupervised methods in the KAD field. Lastly, we select ATAD [41] and AnoTransfer [40], two exemplary models of transfer learning-based anomaly detection, which exhibit capability in addressing universal AD challenges. AnoTransfer is also the state-of-the-art universal AD model.

5.1 Dataset and Evaluation Metric

We conduct experiments on four datasets collected from different real-world online service systems. The overall dataset statistics can be found in Appendix D. Dataset $\mathcal{A}$ is a public dataset from the 2018 International Artificial Intelligence for IT Operations (AIOps) algorithm competition [16]. Dataset $\mathcal{B}$ is Yahoo Webscope collected from Yahoo online service systems [12]. Dataset $\mathcal{C}$ is NAB, a benchmark for evaluating time-series anomaly detection algorithms in real-time applications [13]. Dataset $\mathcal{D}$ is collected from a real-world cloud service systems serving millions of users. Due to the space limitation, more information about the datasets can be found in Appendix D.

Precision, recall, and F1-score (denoted as P, R, and $F1$) are common KPI anomaly detection metrics, but their traditional forms are not ideal for interval anomalies in KAD. Improved metrics have emerged to address this gap and been widely used in KAD area [14, 18, 28, 33, 40] (notated as P^*, R^* and $F1^*$). In this evaluation approach, a labeled anomalous segment is deemed correctly detected if any part is identified, marking it as a true positive or, if overlooked, a false negative. To assess methods holistically, we use both traditional and enhanced metrics, alongside the Area Under the Curve (AUC). Efficiency is gauged by measuring each model's total processing time, from pre-training to inference.

5.2 Overall Performance (RQ1)

To evaluate the overall performance of *KAD-Disformer*, we consecutively choose one out of the four datasets as the target dataset for fine-tuning and testing and the rest three datasets as the source dataset for pre-training. For the selected target dataset, we split it into two parts: the training part (50%) and the test part (50%). As the sufficient data, shown in Table 2, 50% is enough for convergence of each model. For the transferable methods (ATAD, AnoTransfer, and *KAD-Disformer*), we train the model from the source datasets and tune the model on the training part of target dataset and evaluate the model on the test part of target dataset. To test whether *KAD-Disformer* can keep high performance with fewer fine-tuning samples, we use the first 10%, 50% of training part of target dataset to tune *KAD-Disformer* and test *KAD-Disformer* on the test part of target dataset. For the non-transferable methods, we train the models on source dataset and training part of target dataset and test the models on test part of target dataset. It is noteworthy that we also train the non-transferable methods from scratch using the training part of dataset, the results are the same and are omitted for

Table 1: Overall performance of comparative methods. $\mathcal{B},C,\mathcal{D}\rightarrow\mathcal{A}$ indicates that dataset $\mathcal{A}$ is selected as the fine-tuning dataset and $\mathcal{B},C,\mathcal{D}$ are selected as the pre-training datasets, and so forth. *KAD-Disformer*-10%, *KAD-Disformer*-50% and *KAD-Disformer*-100% indicate that we use the first 10%, 50% and 100% of training part of the fine-tuning dataset to tune *KAD-Disformer*. w/o DPM indicates *KAD-Disformer* without Disentangled Projection Matrices. w/o Adap indicates without Adapter Layers. w/o Denoise indicates without Denoising Reconstruction module. A method with * means it has own tailored fine-tuning mechanism.

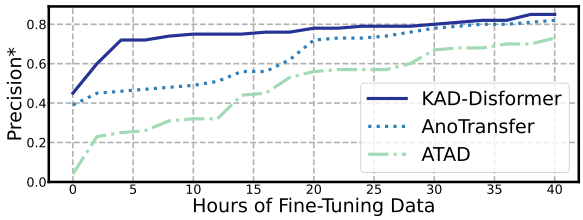
Methods	$\mathcal{B},C,\mathcal{D}\rightarrow\mathcal{A}$						$\mathcal{A},C,\mathcal{D}\rightarrow\mathcal{B}$					
	P^*	R^*	$F1^*$	$F1$	AUC	Time(s)	P^*	R^*	$F1^*$	$F1$	AUC	Time(s)
ARIMA	0.623	0.482	0.543	0.214	0.502	**952**	0.301	0.175	0.237	0.199	0.527	**812**
LSTM-NDT	0.798	0.598	0.683	0.507	0.698	29263	0.643	0.507	0.516	0.498	0.679	27034
Donut	0.779	0.706	0.740	0.602	0.751	15241	0.486	0.737	0.618	0.545	0.715	14024
AnomalyTrans	0.813	0.774	0.792	0.649	0.769	20173	0.704	0.685	0.695	0.600	0.746	18115
ATAD*	0.711	0.770	0.735	0.605	0.736	18743	**0.792**	0.362	0.479	0.301	0.583	28812
AnoTransfer*	0.820	0.733	0.773	0.619	0.760	3921	0.738	0.517	0.623	0.595	0.729	6121
*KAD-Disformer**-10%(w/o DPM)	0.617	0.560	0.587	0.521	0.696	3199	0.412	0.546	0.470	0.324	0.573	5468
*KAD-Disformer** (w/o DPM)	0.801	0.705	0.748	0.640	0.763	3586	0.597	**0.757**	0.656	0.585	0.723	5632
*KAD-Disformer** (w/o Adap)	0.691	0.634	0.661	0.612	0.697	3386	0.566	0.641	0.601	0.580	0.721	5401
*KAD-Disformer** (w/o Denoise)	0.781	0.662	0.692	0.607	0.737	3582	0.688	0.626	0.656	0.582	0.719	5592
*KAD-Disformer**-10%	0.774	0.814	0.793	0.684	0.790	3351	0.593	0.501	0.542	0.517	0.690	5786
*KAD-Disformer**-50%	0.845	**0.935**	0.840	0.704	0.794	3660	0.727	0.658	0.691	0.566	0.724	5897
*KAD-Disformer**-100%	**0.859**	0.890	**0.874**	**0.717**	**0.819**	3893	0.770	0.668	**0.716**	**0.623**	**0.758**	5988

Methods	$\mathcal{A},\mathcal{B},\mathcal{D}\rightarrow C$						$\mathcal{A},\mathcal{B},C\rightarrow\mathcal{D}$					
	P^*	R^*	$F1^*$	$F1$	AUC	Time(s)	P^*	R^*	$F1^*$	$F1$	AUC	Time(s)
ARIMA	0.630	0.613	0.607	0.236	0.519	**736**	0.605	0.552	0.579	0.244	0.561	**980**
LSTM-NDT	0.656	0.727	0.662	0.583	0.717	26566	0.674	0.697	0.684	0.601	0.741	30998
Donut	0.763	0.716	0.762	0.624	0.752	14669	0.718	0.736	0.726	0.640	0.774	15428
AnomalyTrans	0.794	0.927	0.853	0.674	0.798	17769	**0.909**	0.813	0.857	0.685	0.827	21540
ATAD*	0.862	0.767	0.807	0.572	0.715	27903	0.875	0.744	0.82	0.646	0.764	15044
AnoTransfer*	0.852	0.827	0.833	0.592	0.737	6405	0.839	0.798	0.817	0.663	0.774	3175
*KAD-Disformer**-10%(w/o DPM)	0.595	0.639	0.616	0.422	0.639	5612	0.636	0.561	0.596	0.490	0.670	2843
*KAD-Disformer** (w/o DPM)	0.916	0.641	0.754	0.669	0.777	6208	0.785	0.755	0.769	0.664	0.719	3074
*KAD-Disformer** (w/o Adap)	0.699	0.678	0.688	0.591	0.736	5672	0.532	0.707	0.607	0.573	0.720	3002
*KAD-Disformer** (w/o Denoise)	0.782	0.710	0.745	0.609	0.742	5983	0.848	0.821	0.834	0.684	0.802	2977
*KAD-Disformer**-10%	0.809	0.861	0.834	0.601	0.737	5716	0.821	0.858	0.839	0.667	0.766	2914
*KAD-Disformer**-50%	0.814	**0.929**	0.868	0.646	0.762	6021	0.854	0.904	0.878	0.681	0.793	3074
*KAD-Disformer**-100%	**0.944**	0.855	**0.897**	**0.719**	**0.813**	6316	0.856	**0.916**	**0.884**	**0.724**	**0.831**	3114

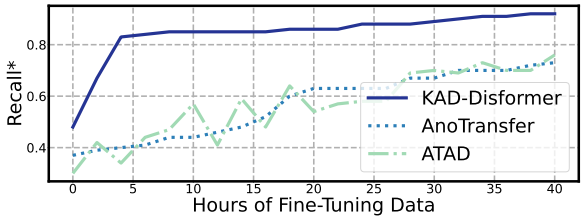
space limitation. Since ATAD is a semi-supervised method, we give partial labels of each time series as its requirements to reproduce the performance in its paper. The results of the experiments are shown in Table 1.

From Table 1, we observe that in all the four target datasets, *KAD-Disformer* achieves the best F1-score and AUC over all comparative methods, including the classic statistic methods and deep learning-based methods. *KAD-Disformer* also achieves the least total time consumption (including pre-training, fine-tuning, and inference) among deep learning-based methods. The improvements in detection accuracy mainly come from two folds. The first is the architecture of the Transformer. This is confirmed by the fact that the Transformer-based methods such as AnomalyTrans and *KAD-Disformer* achieve an improved detection accuracy compared with RNN-based (LSTM-NDT) and VAE-based (Donut) models. Second, denoising reconstruction alleviates the negative influence of noises and anomalies in the training data. The contribution of the denoising reconstruction is analyzed in Appendix F.2. Remarkably, with only 10% fine-tuning data, *KAD-Disformer* achieves a comparable, even better performance than other methods. This is the contribution of our uTune mechanism. The further analysis of uTune is in Section 5.5.

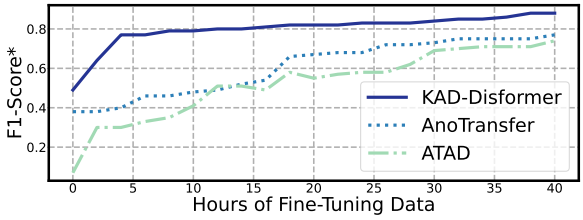
There is no doubt that the classic statistic method ARIMA is the most efficient method. However, the accuracy is the lowest. LSTM-NDT is almost the worst method with respect to time efficiency due to its sequential module LSTM. Besides, due to the existence of early-stop mechanism, the convergence speed of LSTM also limits its efficiency. The higher the convergence speed is, the less time it spends. It is surprising that Donut is quite efficient and spends comparative, even less time compared with the transferable ATAD. This is because of the simple architecture of the model and the noteworthy converging speed of VAE. Another important observation is that AnomalyTrans and ATAD, consume a lot of time on each dataset. For AnomalyTrans, the high-level time consumption is caused by the computation-intensive Transformer-based architecture. For ATAD, the high-level time consumption is caused by the random forest architecture.



(a) Precision



(b) Recall



(c) F1-score

Figure 4: The performance of *KAD-Disformer*, AnoTransfer and ATAD tuned with different percentages of data.

5.3 Few-Shot Learning Ability (RQ2)

Due to our novel disentangled projection matrices and uTune mechanism, *KAD-Disformer* can be well-tuned with a handful of samples, which indicates *KAD-Disformer* can save a lot of time for collecting fine-tuning data. To evaluate the abovementioned properties of *KAD-Disformer*, we evaluate the accuracy of our model varies with different numbers of fine-tuning samples. The results are shown in Figure 4. We gradually give more fine-tuning samples (add 2h data every time) to the model and record the performance metrics. Here we use $\mathcal{B}, C, \mathcal{D} \rightarrow \mathcal{A}$ scenario in this experiment and take ATAD and AnoTransfer as the baseline models. There are two reasons why we select dataset $\mathcal{A}$ as the fine-tuning dataset in this experiment. The first reason is that dataset $\mathcal{A}$ is a public dataset, which is beneficial for reproducing our results. The second reason is that dataset $\mathcal{A}$ has the largest number of data points and longest mean length of the curves among three public datasets $\mathcal{A}$, $\mathcal{B}$ and C, which can cover a wide range of fine-tuning data's size.

From the results, we find that *KAD-Disformer* consistently outperforms the other two comparative methods given the same quantity of data, indicating the high effectiveness of *KAD-Disformer* in quickly adapting from source to target. Another observation we can get from Figure 4 is that AnoTransfer achieves better performance than ATAD, which is consistent with the result of [40]. More importantly, we find that the performance of *KAD-Disformer* grows much faster with the growth of fine-tuning data. With only 1/8 of fine-tuning samples , *KAD-Disformer* achieves competitive performance with AnoTransfer saving about 25 hours. This confirms the fact that **our *KAD-Disformer* has the ability of few-shot learning**. *KAD-Disformer* quickly adapts to an extremely small number of new incoming fine-tuning samples and keeps high-level generalization due to the "learn to learn" mechanism. As a result, *KAD-Disformer* can significantly reduce the time of collecting sufficient fine-tuning data and enables a fast deployment in the real-world environment.

5.4 Ablation Study of Disentangled Projection Matrices (RQ3)

We conduct an ablation study to test the effectiveness of disentangled projection matrices (denoted as DPM). We compare the performance with and without DPM (replace with the original self-attention) under similar experiment settings to Section 5.2. Without DPM, all the projection matrices are updated during fine-tuning.

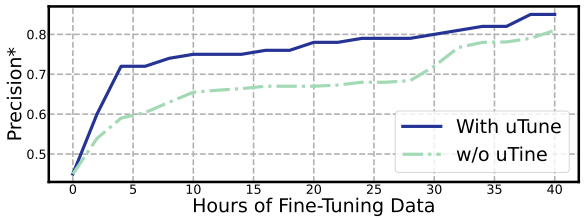
From Table 1, we observe that the model with DPM outperforms the one without DPM. Remarkably, given 10% fine-tuning data, the model without DPM suffers severe performance degradation. We conclude that the improvement brought by DPM comes from three parts. The first is that DPM can save knowledge learned from pre-training in W_{common} and keep them in the fine-tuning stage. Without DPM, all the parameters of projection matrices are updated during fine-tuning, which may lose some common knowledge and lead to performance degradation. The second part is more learnable parameters during fine-tuning for *KAD-Disformer*. Projecting matrices are the most important parameters of Transformer. By disentangling the projection matrices, we get double learnable parameters. Generally speaking, more learnable parameters give the model more potential to achieve better performance. The third part is that DPM alleviates the overfitting problem during fine-tuning thanks to the uTune mechanism illustrated in Section 4.4.

5.5 Contribution of uTune (RQ4)

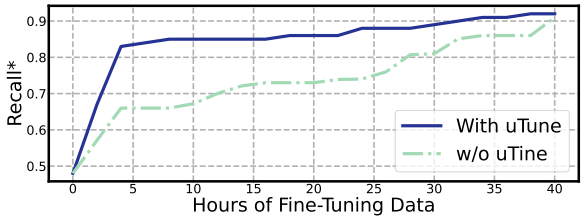
To evaluate the contribution of our uTune mechanism, we conduct an ablation study under a similar setting to Section 5.3. We apply the traditional fine-tuning technique and our uTune to the same pre-trained *KAD-Disformer* model, respectively, and compare the performance tuned with different quantities of fine-tuning data. The result is shown in Figure 5.

From Figure 5, we find that without uTune, the performance suffers apparent degradation when we use small-scale (10%) fine-tuning data. Given 5h of fine-tuning data, the F1-score decreases nearly 30% compared with the model with uTune. Besides, even given full fine-tuning data, the performance of uTune is still 14% higher than the traditional fine-tuning technique.

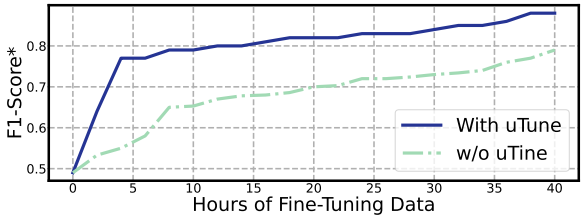
Visualization. To further understand what uTune does, we collect the outputs of *KAD-Disformer*'s encoder and compare the differences between the outputs after pre-training and fine-tuning. We aggregate dataset $\mathcal{A}$, $\mathcal{B}$, C as the pre-train dataset and regard dataset $\mathcal{D}$ as the fine-tuning dataset. We randomly sample 300 data points from the pre-train data and 80 from the fine-tuning data. Then we feed them to the models before and after the uTune. We collect the outputs of the encoder and apply t-SNE [30] to reduce



(a) Precision

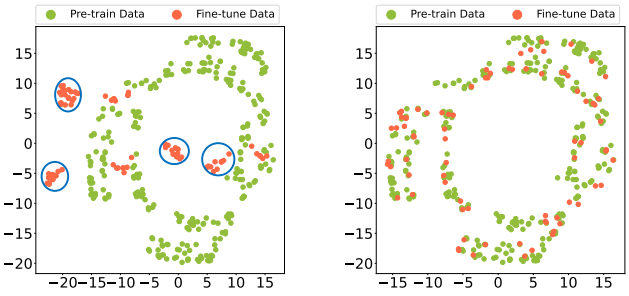


(b) Recall



(c) F1-score

Figure 5: The performance comparison of *KAD-Disformer* with and without uTune mechanism given different percentages of fine-tuning data.



(a) before uTune (b) after uTune

Figure 6: The distribution of the encoder outputs before and after uTune visualized by t-SNE.

the dimension of the concatenated outputs and visualize them in a 2D-scatter figure shown as Figure 6.

The first observation we got from Figure 6 is that the distribution of points of fine-tuning data (red) in Figure 6a is distinct from the distribution of the pre-train data (green). It is reasonable that the pre-trained model, *i.e.,* before uTune, has never seen the unseen KPIs from the fine-tuning data, the encoder could not properly encode these data to the proper position. However, after uTune, the distributions of the data points from fine-tuning KPIs become very similar to the distribution of pre-train data, which is shown in Figure 6b. The result demonstrates that our uTune can effectively adapt the distribution of the fine-tuning data to the pre-train data. After the uTune, the decoders are familiar with the inputs, leading to a satisfying performance.

6 Lessons Learned From Deployment

Deploying *KAD-Disformer* in a real-world cloud service system serving millions of users and integrating Microsoft's online KPI data has offered significant insights [9, 35, 39]. We highlight key takeaways:

Pre-trained KPI. Our experience has led us to conclude that a pre-trained dataset should encompass a broad spectrum of KPI types. This diversity ensures that the dataset's coverage extends across various KPIs, maintaining a balanced ratio in terms of the volume of data per KPI type. Additionally, it is paramount that the dataset spans an extensive period ranging from several days to multiple months. Such a temporal breadth is crucial for fostering a robust generalization capability during the pre-training phase.

Fine-tuning Phase. Our findings underscore the importance of the data length provided for fine-tuning. Ideally, this data should comprehensively cover an entire cycle to ensure that the fine-tuning process is as effective as possible. We observed that when the fine-tuning data exceeded the span of one cycle, there was a notable enhancement in the system's performance after fine-tuning.

7 Conclusion

Universal KPI anomaly detection is a crucial but challenging task for large-scale online service systems with hundreds of millions of KPIs. In this paper, we propose a disentangled transformer model named *KAD-Disformer* to efficiently and effectively detect anomalies for an enormous number of KPIs. We novelly design the disentangled projection matrices and uTune mechanism to help the model quickly fit the incoming KPI with limited fine-tuning samples without the risk of over-fitting. Besides, the Denoising Reconstruction technique can alleviate the influence of noises and make our *KAD-Disformer* more robust. We conduct experiments on four different real-world datasets, and the results show that *KAD-Disformer* outperforms the current state-of-the-art universal anomaly detection model by 13% in F1-score and achieves comparable performance with only 1/8 of fine-tuning samples saving about 25 hours. *KAD-Disformer* has been deployed in a real-world online service system serving millions of people for months. Besides, we are glad to share the source code of *KAD-Disformer* for researchers and engineers in this area. Our code is available at https://github.com/NetManAIOps/KAD-Disformer.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant No.62072264 and No.62202445, the National Key Research and Development Program of China No.2022YFB2901800, and the Beijing National Research Center for Information Science and Technology (BNRist) key projects.

References

[1] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[2] Chris Chatfield and Mohammad Yar. 1988. Holt-Winters forecasting: some practical issues. *Journal of the Royal Statistical Society: Series D (The Statistician)* 37, 2 (1988), 129–140.

[3] Wenxiao Chen, Haowen Xu, Zeyan Li, Dan Pei, Jie Chen, Honglin Qiao, Yang Feng, and Zhaogang Wang. 2019. Unsupervised Anomaly Detection for Intricate KPIs via Adversarial Training of VAE. In *2019 IEEE Conference on Computer Communications, INFOCOM 2019, Paris, France, April 29 - May 2, 2019*. IEEE, 1891–1899.

[4] Xuanhao Chen, Liwei Deng, Feiteng Huang, Chengwei Zhang, Zongquan Zhang, Yan Zhao, and Kai Zheng. 2021. DAEMON: Unsupervised Anomaly Detection and Interpretation for Multivariate Time Series. In *ICDE*. IEEE, 2225–2230.

[5] Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, Jun Zeng, Supriyo Ghosh, Xuchao Zhang, Chaoyun Zhang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Tianyin Xu. 2024. Automatic Root Cause Analysis via Large Language Models for Cloud Incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems, EuroSys 2024, Athens, Greece, April 22-25, 2024*. ACM, 674–688. https://doi.org/10.1145/3627703.3629553

[6] Zhuangbin Chen, Jinyang Liu, Yuxin Su, Hongyu Zhang, Xiao Ling, and Michael R. Lyu. 2022. Adaptive Performance Anomaly Detection for Online Service Systems via Pattern Sketching. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*. IEEE, 61–72.

[7] 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.

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*. PMLR, 1126–1135.

[9] Vaibhav Ganatra, Anjaly Parayil, Supriyo Ghosh, Yu Kang, Minghua Ma, Chetan Bansal, Suman Nath, and Jonathan Mace. 2023. Detection Is Better Than Cure: A Cloud Incidents Perspective. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023, San Francisco, CA, USA, December 3-9, 2023*, Satish Chandra, Kelly Blincoe, and Paolo Tonella (Eds.). ACM, 1891–1902. https://doi.org/10.1145/3611643.3613898

[10] Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. 2021. On the Effectiveness of Adapter-based Tuning for Pretrained Language Model Adaptation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, 2208–2222.

[11] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Söderström. 2018. Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Yike Guo and Faisal Farooq (Eds.). ACM, 387–395.

[12] Nikolay Laptev, Saeed Amizadeh, and Ian Flint. 2015. Generic and Scalable Framework for Automated Time-series Anomaly Detection. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, Longbing Cao, Chengqi Zhang, Thorsten Joachims, Geoffrey I. Webb, Dragos D. Margineantu, and Graham Williams (Eds.). ACM, 1939–1947.

[13] Alexander Lavin and Subutai Ahmad. 2015. Evaluating Real-Time Anomaly Detection Algorithms - The Numenta Anomaly Benchmark. In *14th IEEE International Conference on Machine Learning and Applications, ICMLA 2015, Miami, FL, USA, December 9-11, 2015*, Tao Li, Lukasz A. Kurgan, Vasile Palade, Randy Goebel, Andreas Holzinger, Karin Verspoor, and M. Arif Wani (Eds.). IEEE, 38–44.

[14] Jia Li, Shimin Di, Yanyan Shen, and Lei Chen. 2021. FluxEV: A Fast and Effective Unsupervised Framework for Time-Series Anomaly Detection. In *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*, Liane Lewin-Eytan, David Carmel, Elad Yom-Tov, Eugene Agichtein, and Evgeniy Gabrilovich (Eds.). ACM, 824–832.

[15] Zeyan Li, Wenxiao Chen, and Dan Pei. 2018. Robust and Unsupervised KPI Anomaly Detection Based on Conditional Variational Autoencoder. In *37th IEEE International Performance Computing and Communications Conference, IPCCC 2018, Orlando, FL, USA, November 17-19, 2018*. IEEE, 1–9.

[16] Zeyan Li, Nengwen Zhao, Shenglin Zhang, Yongqian Sun, Pengfei Chen, Xidao Wen, Minghua Ma, and Dan Pei. 2022. Constructing Large-Scale Real-World Benchmark Datasets for AIOps. *arXiv preprint arXiv:2208.03938* (2022).

[17] 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 J. Sel. Areas Commun.* 40, 9 (2022), 2746–2765.

[18] 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 *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, Feida Zhu, Beng Chin Ooi, and Chunyan Miao (Eds.). ACM, 3220–3230.

[19] Dapeng Liu, Youjian Zhao, Haowen Xu, Yongqian Sun, Dan Pei, Jiao Luo, Xiaowei Jing, and Mei Feng. 2015. Opprentice: Towards Practical and Automatic Anomaly Detection Through Machine Learning. In *Proceedings of the 2015 ACM Internet Measurement Conference, IMC 2015, Tokyo, Japan, October 28-30, 2015*, Kenjiro Cho, Kensuke Fukuda, Vivek S. Pai, and Neil Spring (Eds.). ACM, 211–224.

[20] 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 *2021 USENIX Annual Technical Conference, USENIX ATC 2021, July 14-16, 2021*, Irina Calciu and Geoff Kuenning (Eds.). USENIX Association, 413–426. https://www.usenix.org/conference/atc21/presentation/ma

[21] 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 *29th IEEE International Symposium on Software Reliability Engineering, ISSRE 2018, Memphis, TN, USA, October 15-18, 2018*, Sudipto Ghosh, Roberto Natella, Bojan Cukic, Robin S. Poston, and Nuno Laranjeiro (Eds.). IEEE Computer Society, 13–24. https://doi.org/10.1109/ISSRE.2018.00013

[22] H Zare Moayedi and MA Masnadi-Shirazi. 2008. Arima model for network traffic prediction and anomaly detection. In *2008 international symposium on information technology*, Vol. 4. IEEE, 1–6.

[23] Henri J Nussbaumer. 1981. The fast Fourier transform. In *Fast Fourier Transform and Convolution Algorithms*. Springer, 80–111.

[24] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[25] Panpan Qi, Dan Li, and See-Kiong Ng. 2022. MAD-SGCN: Multivariate Anomaly Detection with Self-learning Graph Convolutional Networks. In *ICDE*. IEEE, 1232–1244.

[26] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. 2019. Time-Series Anomaly Detection Service at Microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis (Eds.). ACM, 3009–3017.

[27] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. 2022. Anomaly Detection in Time Series: A Comprehensive Evaluation. *Proc. VLDB Endow.* 15, 9 (2022), 1779–1797.

[28] 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, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis (Eds.). ACM, 2828–2837.

[29] Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. 2022. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data. *Proc. VLDB Endow.* 15, 6 (2022), 1201–1214.

[30] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008.

[32] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems* 34 (2021), 22419–22430.

[33] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, Jie Chen, Zhaogang Wang, and Honglin Qiao. 2018. Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, Pierre-Antoine Champin, Fabien Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis (Eds.). ACM, 187–196.

[34] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2021. Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy. *CoRR* abs/2110.02642 (2021). arXiv:2110.02642

[35] Zhaoyang Yu, Minghua Ma, Chaoyun Zhang, Si Qin, Yu Kang, Chetan Bansal, Saravan Rajmohan, Yingnong Dang, Changhua Pei, Dan Pei, Qingwei Lin, and Dongmei Zhang. 2024. MonitorAssistant: Simplifying Cloud Service Monitoring via Large Language Models. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*.

[36] Zhaoyang Yu, Qianyu Ouyang, Changhua Pei, Xin Wang, Wenxiao Chen, Liangfei Su, Huai Jiang, Xuanrun Wang, Jianhui Li, and Dan Pei. 2024. Causality Enhanced Graph Representation Learning for Alert-Based Root Cause Analysis. In *CCGrid*. IEEE.

[37] Zhaoyang Yu, Changhua Pei, Shenglin Zhang, Xidao Wen, Jianhui Li, Gaogang Xie, and Dan Pei. 2023. AutoKAD: Empowering KPI Anomaly Detection with Label-Free Deployment. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 13–23.

[38] Zhaoyang Yu, Shenglin Zhang, Mingze Sun, Yingke Li, Yankai Zhao, Xiaolei Hua, Lin Zhu, Xidao Wen, and Dan Pei. 2024. Supervised Fine-Tuning for Unsupervised KPI Anomaly Detection for Mobile Web Systems. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, Tat-Seng Chua, Chong-Wah Ngo, Ravi Kumar, Hady W. Lauw, and Roy Ka-Wei Lee (Eds.). ACM, 2859–2869. https://doi.org/10.1145/3589334.3645392

[39] Zhengran Zeng, Yuqun Zhang, Yong Xu, Minghua Ma, Bo Qiao, Wentao Zou, Qingjun Chen, Meng Zhang, Xu Zhang, Hongyu Zhang, Xuedong Gao, Hao Fan, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. TraceArk: Towards Actionable Performance Anomaly Alerting for Online Service Systems. In *45th IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice, SEIP@ICSE 2023, Melbourne, Australia, May 14-20, 2023*. IEEE, 258–269. https://doi.org/10.1109/ICSE-SEIP58684.2023.00029

[40] Shenglin Zhang, Zhenyu Zhong, Dongwen Li, Qiliang Fan, Yongqian Sun, Man Zhu, Yuzhi Zhang, Dan Pei, Jiyan Sun, Yinlong Liu, Hui Yang, and Yongqiang Zou. 2022. Efficient KPI Anomaly Detection Through Transfer Learning for Large-Scale Web Services. *IEEE J. Sel. Areas Commun.* 40, 8 (2022), 2440–2455. https://doi.org/10.1109/JSAC.2022.3180785

[41] Xu Zhang, Qingwei Lin, Yong Xu, Si Qin, Hongyu Zhang, Bo Qiao, Yingnong Dang, Xinsheng Yang, Qian Cheng, Murali Chintalapati, Youjiang Wu, Ken Hsieh, Kaixin Sui, Xin Meng, Yaohai Xu, Wenchi Zhang, Furao Shen, and Dongmei Zhang. 2019. Cross-dataset Time Series Anomaly Detection for Cloud Systems. In *2019 USENIX Annual Technical Conference, USENIX ATC 2019, Renton, WA, USA, July 10-12, 2019*, Dahlia Malkhi and Dan Tsafrir (Eds.). USENIX Association, 1063–1076.

[42] 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 *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021*, Diomidis Spinellis, Georgios Gousios, Marsha Chechik, and Massimiliano Di Penta (Eds.). ACM, 527–539. https://doi.org/10.1145/3468264.3468543

[43] Zhenyu Zhong, Qiliang Fan, Jiacheng Zhang, Minghua Ma, Shenglin Zhang, Yongqian Sun, Qingwei Lin, Yuzhi Zhang, and Dan Pei. 2023. A Survey of Time Series Anomaly Detection Methods in the AIOps Domain. *CoRR* abs/2308.00393 (2023). https://doi.org/10.48550/ARXIV.2308.00393 arXiv:2308.00393

A Experiment Environment

For comparing the models' efficiency, all the experiments are conducted on a single node server. The CPU is an Intel(R) Xeon(R) Gold 5218R with 128G memory, the GPUs are four NVIDIA GeForce RTX 3090, which is a common setup for Internet companies.

B Pseudo code for workflows

The workflow of the fine-tuning (uTune) is shown in Algorithm 2 The workflow of the pre-training is shown in Algorithm 1

Algorithm 1: Pre-train workflow

Input : $D_S = \{X_0^s, X_1^s, X_2^s, \cdots, X_N^s\}$: KPI datasets for pre-training
Input : θ: all parameters of the model (including disentangled projection matrices and adapter layers)
Input : f_θ: initial model
Input : lr: learning rate

Randomly initialize θ
Preprocess each KPI in D_S
while *not done* **do**
  Randomly sample a mini-batch x from D_S
  Evaluate loss $L(x; f_\theta)$ on x
  Update $\theta \leftarrow \theta - lr \cdot \frac{\partial L}{\partial \theta}$
end
Output : f_θ: the pre-trained model

Algorithm 2: uTune

Input : f_θ: the pre-trained based model
Input : X: KPI data for fine-tuning
Input : D_S: KPI dataset for pre-training
Input : $w_{personal}$: the parameters of personal projection matrices
Input : w_{adap}: the parameters of adapter layers
Input : $\alpha \in [0, 1]$: hyper-parameter for meta loss
Input : lr: learning rate

Preprocess KPI X
while *not done* **do**
  Randomly sample a mini-batch x_1 from X
  Randomly sample a mini-batch x_2 from D_S
  Evaluate loss $L_1 = L(x_1, f_\theta)$ on x_1
  Update $w_{personal} \leftarrow w_{personal} - lr \cdot \frac{\partial L_1}{\partial w_{personal}}$ and $w_{adap} \leftarrow w_{adap} - lr \cdot \frac{\partial L_1}{\partial w_{adap}}$ to get an updated model f_θ^*
  Evaluate loss $L_2 = L(x_2, f_\theta^*)$ on x_2
  Compute meta loss $L = \alpha L_1 + (1 - \alpha) L_2$
  Update $w_{personal} \leftarrow w_{personal} - lr \cdot \frac{\partial L}{\partial w_{personal}}$ and $w_{adap} \leftarrow w_{adap} - lr \cdot \frac{\partial L}{\partial w_{adap}}$
end
Output : f_θ^*: the fine-tuned model

C Sensitivity analysis of Hyper-Parameters

We use the setting of $\mathcal{B}, \mathcal{C}, \mathcal{D} \rightarrow \mathcal{A}$ to analyze the influence of two hyper-parameters: α in uTune and the number of the stacked encoder and encoder layers N in *KAD-Disformer*.

From the result, we find that when $\alpha \in [0.2, 0.7], N \geq 3$, the performance of *KAD-Disformer* is stable and satisfying. Thus, in our experiment, we choose $\alpha = 0.5, N = 3$.

D Dataset Description

Dataset $\mathcal{A}$ comprises 29 time series from various Internet companies, labeled by domain experts. These series, representing metrics like response time and network latency, demonstrate the method's real-world performance.

Dataset $\mathcal{B}$ includes real-world and synthetic time series from Yahoo's online services, each tagged with various anomaly types.

Dataset $\mathcal{C}$ consists of sub-datasets from companies like Twitter and AWS, containing time series of different lengths and metrics such as CPU utilization and cost-per-click.

Dataset $\mathcal{D}$ features 67 time series from a cloud service system in Microsoft, spanning three months with minute-level granularity, labeled by experienced operators to reflect system health.

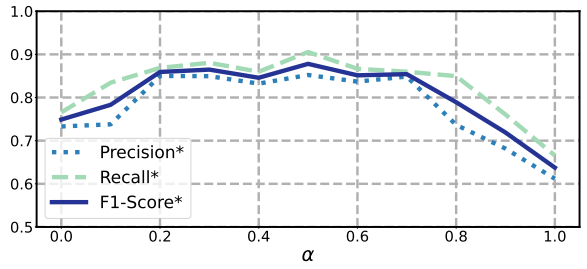


Figure 7: The performance of *KAD-Disformer* with different alpha.

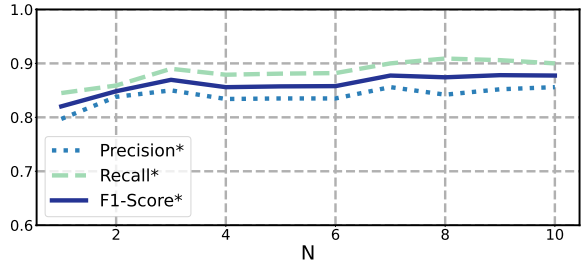


Figure 8: The performance of *KAD-Disformer* with different numbers of encoder and decoder layers (N).

Table 2: The statistics of each dataset. $\mathcal{A}, \mathcal{B}, C$ are all public dataset and $\mathcal{D}$ is collected from a real-world web service provider.

Dataset	#Curves	Mean length	#Points	%Anomaly
$\mathcal{A}$ (IOPS [16])	29	204238	5922913	2.264%
$\mathcal{B}$ (Yahoo[12])	367	1561	572966	0.683%
C (NAB [13])	52	6565	341366	9.812%
$\mathcal{D}$ (Industry)	67	111094	7443327	0.922%

E Evaluation Metric

Improved anomaly detection metrics have recently been introduced and applied to contemporary research [14, 26, 33]. Consider a labeled, continuous anomaly segment: we categorize the segment as accurately detected if the algorithm identifies any anomaly within that segment. Thus, every point within this anomalous segment is designated as a true positive (TP). Conversely, if the model fails to identify an anomaly, every point within the segment is designated a false negative (FN). Points that lie outside these abnormal segments are not adjusted.

F More Ablation Study

F.1 Ablation Study of Disentangled Projection Matrices

The full version of the performance comparison between our approach with and without disentangled projection matrices (DPM) is shown in Table 3.

F.2 Ablation Study of Adapter Layers and Denoising Reconstruction Module

To verify the contributions of the adapter layers and denoising reconstruction module, we conduct an ablation study by removing the adapter layers and denoising decoder respectively. The comparison results are shown in Table 3.

The results show that the performance decreases after removing either adapter layers or denoising the reconstruction module. Without adapter layers, the F1 decreases up to 32%, and without denoising reconstruction, the F1 decreases up to 20%. However, the reduction of time is marginal. Similar to DPM, the adapter layers are tunable parameters of *KAD-Disformer*, which directly decides the capacity of fine-tuning. The denoising reconstruction can alleviate the influence of noise and anomalies in the training data, avoiding *KAD-Disformer* from learning abnormal patterns, which is also confirmed by the previous work Donut [33].

Table 3: Performance Comparison with Different Configurations

Methods	$\mathcal{B}, C, \mathcal{D} \rightarrow \mathcal{A}$					
	P^*	R^*	$F1^*$	$F1$	AUC	Time(s)
w/o Adap	0.691	0.634	0.661	0.612	0.697	3386
w/o Denoise	0.781	0.662	0.692	0.607	0.737	3582
w/o DPM-10%	0.617	0.560	0.587	0.521	0.696	**3199**
w/o DPM	0.801	0.705	0.748	0.640	0.763	3586
KAD-Disformer-10%	0.774	0.814	0.793	0.684	0.790	3351
KAD-Disformer	**0.859**	**0.890**	**0.874**	**0.717**	**0.819**	3893
Methods	$\mathcal{A}, C, \mathcal{D} \rightarrow \mathcal{B}$					
	P^*	R^*	$F1^*$	$F1$	AUC	Time(s)
w/o Adap	0.566	0.641	0.601	0.580	0.721	**5401**
w/o Denoise	0.688	0.626	0.656	0.582	0.719	5592
w/o DPM-10%	0.412	0.546	0.470	0.324	0.573	5468
w/o DPM	0.597	**0.757**	0.656	0.585	0.723	5632
KAD-Disformer-10%	0.593	0.501	0.542	0.517	0.690	5786
KAD-Disformer	**0.770**	0.668	**0.716**	**0.623**	**0.758**	5988
Methods	$\mathcal{A}, \mathcal{B}, \mathcal{D} \rightarrow C$					
	P^*	R^*	$F1^*$	$F1$	AUC	Time(s)
w/o Adap	0.699	0.678	0.688	0.591	0.736	5672
w/o Denoise	0.782	0.710	0.745	0.609	0.742	5983
w/o DPM-10%	0.595	0.639	0.616	0.422	0.639	**5612**
w/o DPM	0.916	0.641	0.754	0.669	0.777	6208
KAD-Disformer-10%	0.809	**0.861**	0.834	0.601	0.737	5716
KAD-Disformer	**0.944**	0.855	**0.897**	**0.719**	**0.813**	6316
Methods	$\mathcal{A}, \mathcal{B}, C \rightarrow \mathcal{D}$					
	P^*	R^*	$F1^*$	$F1$	AUC	Time(s)
w/o Adap	0.532	0.707	0.607	0.573	0.720	3002
w/o Denoise	0.848	0.821	0.834	0.684	0.802	2977
w/o DPM-10%	0.636	0.561	0.596	0.490	0.670	**2843**
w/o DPM	0.785	0.755	0.769	0.664	0.719	3074
KAD-Disformer-10%	0.821	0.858	0.839	0.667	0.766	2914
KAD-Disformer	**0.856**	**0.916**	**0.884**	**0.724**	**0.831**	3114