Share or Not Share? Towards the Practicability of Deep Models for Unsupervised Anomaly Detection in Modern Online Systems

Zilong He[†], Pengfei Chen[*], Tao Huang[‡][§]
[*][†]*Sun Yat-sen University*, [‡]*Tencent* [§]*Illinois Institute of Technology*
[†]hezlong@mail2.sysu.edu.cn, [*]chenpf7@mail.sysu.edu.cn, [§]thuang21@hawk.iit.edu

***Abstract*—Anomaly detection is crucial in the management of modern online systems. Due to the complexity of patterns in the monitoring data and the lack of labelled data with anomalies, recent studies mainly adopt deep unsupervised models to address this problem. Notably, even though these models have achieved a great success on experimental datasets, there are still several challenges for them to be successfully applied in a real-world modern online system. Such challenges stem from some significant properties of modern online systems, e.g., large scale, diversity and dynamics. This study investigates how these properties affect the adoption of deep anomaly detectors in modern online systems. Furthermore, we claim that model sharing is an effective way to overcome these challenges. To support this claim, we systematically study the feasibility and necessity of model sharing for unsupervised anomaly detection. In addition, we further propose a novel model, Uni-AD, which works well for model sharing. Based upon Transformer encoder layers and Base layers, Uni-AD can effectively model diverse patterns for different monitored entities and further perform anomaly detection accurately. Besides, it can accept variable-length inputs, which is a required property for a model that needs to be shared. Extensive experiments on two real-world large-scale datasets demonstrate the effectiveness and practicality of Uni-AD.**

***Index Terms*—anomaly detection, neural networks, model sharing, online systems**

I. Introduction

With the tremendous growth of user demand and the rapid development of software technologies, such as microservice [1], [2], modern online systems, e.g., e-commerce systems, are undergoing an ever-growing workload and exhibiting an increasing scale and complexity. For example, WeChat system, a typical modern online system, accommodates more than 3000 services running on over 20000 machines [3]. As a result, these systems are super difficult to operate and maintain.

Metrics and Monitored Entities. Typically, to ensure the availability of an online system, on-call engineers need to continuously collect and monitor a large amount of metrics (e.g., Page View latency, CPU usage). Specifically, *metrics* are telemetry data that are recording the real-time status of some monitored entities from an online system. Here, *entities* denote some components which are being monitored. Examples of monitored entities include services, machines, etc. Generally, an issue (e.g., a performance issue) affecting an online system can expose clues in some metrics from a couple of monitored entities. During analysis, these metrics can be aggregated into multivariate time series and provide insightful information of the system. Therefore, automatic anomaly detection on such multivariate time series data plays an important role in the management of modern online systems.

Customized Anomaly Detection vs. Generic Anomaly Detection. To deploy an anomaly detector for an online system, two types of methods can be considered, namely customized methods and generic methods. Customized methods stand for methods that specifically designed by engineers for some scenarios [4], [5]. Such usage-scenario-restricted methods are usually effective, but they rely heavily on the expertise of engineers, so usually they are only used in some safety-critical scenarios. For the system availability monitoring, to lower the barrier of designing customized anomaly detection methods, generic methods are developed. For these methods, to be out-of-the-box is one of the design principles. Towards this design principle, historical metric data are usually used to train a data-driven model to discriminate anomalous patterns from normal ones. Besides, to be unsupervised is another important design principle for generic methods in most cases. This is because high-quality training data with labels are usually out of reach in practice. Plenty of literature in recent years falls into methods of this type [6]–[18] and many of them [7]–[17], [19] adopt deep learning to perform anomaly detection.

Challenges for Deep Anomaly Detection in Modern Online Systems. Though deep anomaly detectors achieve promising results in laboratory experiments recently, some significant properties of modern online systems still pose several challenges for these methods to be successfully applied in practice. Firstly, modern online systems are increasing in scale and complexity, resulting in a tremendous amount of monitored entities. Because different entities may have different behavior patterns, a previous common view for unsupervised anomaly detection in online systems is that each entity should be assigned a corresponding model, which is trained with corresponding historical data [12], [14], [16]. However, since deep models are intrinsically heavy-weighted, with thousands of network parameters, using the same number of models as the number of monitored entities can hinder a successful adoption of these models due to the huge cost of model storing and scheduling. What is more, the requirement of scalability and fault tolerance renders modern online systems highly dynamic. For example, containers in a cloud native application may be frequently created and destroyed to cope with a changing workload. In this situation, acquiring enough historical data from each monitored entity to train a specific model for each of them is extremely difficult. In general, the large scale and dynamics of today's systems give rise to a gap between previous deep anomaly detection methods and their portable deployment in practice.

Motivation. The motivation of this study is to explore a more practical way to use deep unsupervised anomaly detection during the availability monitoring for a modern online system. Actually, the main culprit of the aforementioned gap is that though the deep anomaly detectors will be used in a generic setting, they are not yet designed and evaluated for this setting up to now. The previously applied experimental setting, namely compulsorily training a corresponding deep model for each monitored entity [12], [14], [16], does not fit in a generic setting, thus not practical. Some recent studies [17], [20], [21] elaborate clustering to cope with this problem through dividing entities into different clusters according to their historical data and assigning a model to each cluster. However, these methods suffer a low effectiveness in a dynamic system where entities may be dynamically created with no corresponding historical data ready to match a corresponding cluster. Besides, an additional clustering process can introduce more hyper-parameters, making the joint optimization of the clustering and the anomaly detection model training more complex and thus more difficult to tune well. In this study, we propose to solve the practicality problem by directly sharing a unified deep unsupervised anomaly detection model across multiple monitored entities from the same system. Moreover, we identify designs in previous models which may make them inappropriate for sharing, and further propose a novel model on the basis of Transformer encoder layers and Base layers to help effectively perform anomaly detection with a shared model. Our model can achieve a relative improvement of 6.66%$\sim$77.71% compared with state-of-the-art deep anomaly detection models. Moreover, it is light-weighted and efficient, with an average detection time at a millisecond level.

The contributions of this study are summarized as follows:

- To the best of our knowledge, we are the first to detailedly discuss whether and how to share a unified deep unsupervised anomaly detection model for monitored entities from the same online system. We conduct comprehensive studies to answer why we can and need to perform model sharing for anomaly detection in modern online systems. The observations can inspire a broad range of data-driven research in this field.
- We propose a novel model Uni-AD (Unified Anomaly Detector), which works well for unsupervised anomaly detection under a model-sharing setting. Uni-AD can well model diverse patterns in different entities and perform anomaly detection effectively. Besides, it can directly handle variable-length inputs, which further increases its practicality.
- We conduct extensive experiments to validate Uni-AD on two real-world large-scale datasets (each contains 500 or more sub datasets). Experiment results demonstrate the effectiveness and practicality of Uni-AD[1].

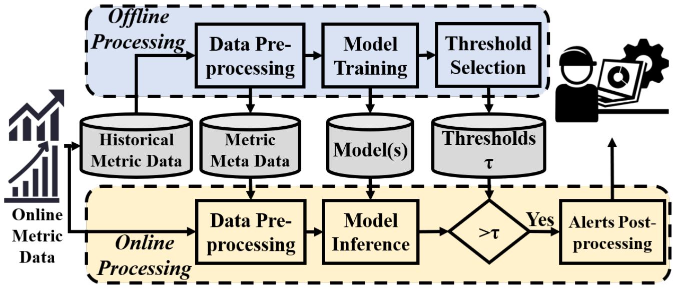


Fig. 1: The overall workflow of deep unsupervised anomaly detection methods in online systems

II. Problem Formulation

Metrics of a monitored entity are formed as multivariate time series. Therefore, anomaly detection in modern online systems is generally a task of multivariate time series anomaly detection. As a common practice, a sliding window is applied over the metric data to gain observations for analysis. An observation $\mathbf{X} \in \mathbb{R}^{N\times W}$ contains metrics of an entity at consecutive timestamps, where N is the number of metrics and W is the length of the sliding window. Each observation can reveal the state of a corresponding entity. We represent the set of entities as $\mathcal{E}$.

Fig. 1 presents a typical workflow for anomaly detection in online systems. The method proposed in this study also follows this workflow. Specifically, the online anomaly detection on an observation $\mathbf{X}$ can be formalized as a function f. This function takes $\mathbf{X}$ as an input. Then after the process of data pre-processing, the anomaly detection model $\mathbf{\Theta}$, a trainable part in f, calculates an anomaly score $s \in \mathbb{R}$ for the input $\mathbf{X}$. This model can be trained offline using historical metric data (e.g., forcing it to learn to reconstruct the input using a compressed representation). Finally, after comparing s to a threshold τ, f decides whether to trigger an alert for the observation $\mathbf{X}$. In a large-scale online system, at each timestamp, the number of observations from different entities, i.e., $|\mathcal{E}|$, can be extremely huge. This study focuses on the practicality of deep models for anomaly detection in this situation.

In the following, we denote the setting of sharing a unified deep unsupervised anomaly detection model $\mathbf{\Theta}$ across various monitored entities as a **model-sharing** setting, and denote the setting of training and using a set of models $\{\mathbf{\Theta}(e)|e \in \mathcal{E}\}$, each corresponding to one specific monitored entity respectively, as a one-for-one setting.

III. Study on Model Sharing

A. Background and Analysis

1) Background Supporting the Feasibility of Model Sharing: Though the total amount of monitored entities may be

[1]The code is in https://github.com/IntelligentDDS/Uni-AD.

large and their functions may be different, the hidden regular patterns of these entities actually will not be too diffused since they all come from the same system. This is because the workloads of different monitored entities from the same system usually exhibit a high correlation due to the replica mechanism and the intrinsic interactions in modern online systems, resulting in a considerable pattern repeatability among them. We will develop some related concepts and conduct extensive experiments to expose such pattern repeatability in Section III-D1.

At the same time, deep models have a prominent capability in modelling various patterns in complex data, which also impels them to maintain their effect when the patterns of the input data are diverse but not totally diffused. Some recent studies [22], [23] have discussed this phenomenon, which will be briefly introduced in Section VI). Besides, through training data sharing, model sharing can act as a type of data augmentation. Specifically, data augmentation refers to a technique which helps avoid over-fitting by generating synthetic data that cover the unexplored input space but remain the original meaning [24]. This technique can effectively enhance the quality and size of the training data, and it is especially useful for deep models, which exactly requires a massive amount of training data. Specifically in our scenario, since different entities are likely to share a similar or closely related generation process (e.g., different instances of a service), the training data from different entities can act as augmented data for each other, which can lighten the adverse effect brought by the over-fitting, and ultimately boost the efficacy of the shared model. In Section III-D2, we will provide an experimental study to show that various deep unsupervised anomaly detection models do still maintain or even boost their effectiveness under a model-sharing setting.

2) Background Implying the Necessity of Model Sharing: The necessity of model sharing stems from the large scale and dynamics of modern online systems. In practice, during online inference, only new coming data need processing, and the processing needs to be conducted along the entity dimension directly (i.e., there is one and only one observation from each entity). By contrast, offline inference is usually first conducted along the time dimension (i.e., many historical observations from each entity) and thus can enjoy the efficiency advantages brought by batch processing implemented through the vectorized execution. However, during online inference under a one-for-one setting, there are many models, and at each time, for each model, only one observation needs to be processed, leading to the unavailability of batch processing. This can give rise to an efficiency gap between online inference and offline inference. If such a gap is not taken into account, the per-sample efficiency estimated in offline experiments can hardly be reproduced online. As a result, whether a deep unsupervised anomaly detection model is efficient enough in practice needs a careful reconsideration, especially when the target is a large-scale system.

Compared with a one-for-one setting, the online model inference with a model-sharing setting can be more efficient for three reasons. First, when a model-sharing setting is adopted and only a single model is on duty, the vectorized execution (i.e., calculating anomaly scores for observations from multiple monitored entities) is well implemented by a majority of deep learning frameworks (e.g., PyTorch [25] and TensorFlow [26]), which makes full use of the AVX instruction set in CPU or the SIMD techniques in GPU and provides an easy-to-use interface for efficient model inference on batches. Second, the memory access load can be reduced, because when the model is shared, it is more likely to be stored in a cache, reducing frequent memory access. Last but not least, the disk IO time for model fetching is also decreased, because when a model-sharing setting is adopted, the total number and size of models are effectively controlled, rendering model reloading unnecessary.

In terms of the dynamics, gaining enough historical data from each entity, which may be newly created, to train a corresponding model is usually unachievable. While if the model is shared, it can be used by all entities. The shared model is able to recognize the pattern of a newly created entity since there usually exist entities with a similar pattern in the training dataset due to the pattern repeatability discussed in Section III-A1.

B. Research Questions

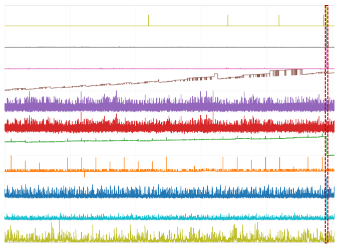
To confirm the analysis above, we collect datasets and conduct experiments to investigate the following questions.

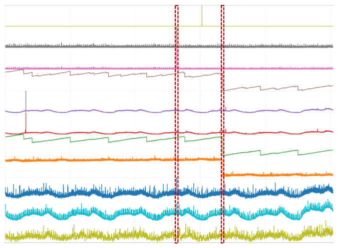
- **RQ1. Feasibility:**
  - **RQ1-1:** How is the repeatability of entity patterns in modern online systems?
  - **RQ1-2:** Can deep anomaly detection models maintain or boost their efficacy when the model is shared across entities in an online system?
- **RQ2. Necessity:** To what degree can model sharing boost the model inference efficiency and bring the adoption of deep anomaly detectors in large-scale systems into practice?

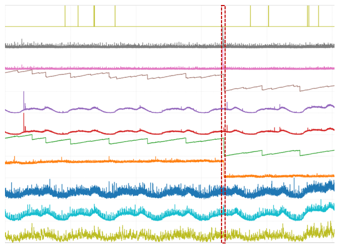
C. Datasets and Environment

We use metric data collected from modern online systems as the datasets to conduct the experimental study. The first dataset, namely CTF_data[2] [17], is the largest publicly available dataset for anomaly detection in an online system. It contains more than 500 entities from a large-scale system. Another large-scale anomaly detection dataset, namely TC_data, is collected from the Tencent Game Services. TC_data contains machine monitoring metrics such as the CPU usage, the memory usage and the network speed. Anomalies in TC_data are labelled according to information from incident tickets. TC_data includes 500 entities. To the best of our knowledge, these two datasets are the most large-scale datasets for anomaly detection on metric data. There also exist some other famous datasets for anomaly detection on metric data,

[2]Some files in its open-sourced repository are missed, so we select the entities without missing files and only use them for the experiments.


(a) IP 192


(b) IP 459


(c) IP 472

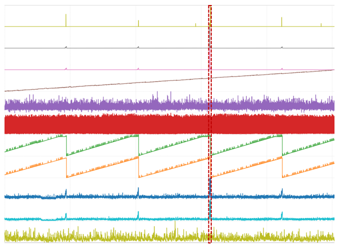

(d) IP 526

Fig. 2: The visualization of some entities in TC_data. Data in the former two days constitute the training dataset, and data in the following five days constitute the testing dataset. Anomalies in the testing dataset are marked in red frames in the figure.

TABLE I: Statistics of the datasets used in this study

Statistics	CTF_data	TC_data	JS_D2+D3	SMD
#entities	525	500	60	28
#metrics / entity	49	11	19	38
#observations	1.97E+7	5.04*E+6	7.78E+5	1.42E+6
anomaly ratio (%)	5.64	0.22	12.75	4.16
train length (day)	5	2	18	14∼18
test length (day)	8	5	25	17∼20

e.g., SMD [12] and JS_D2+D3 [18]. However, SMD and JS_D2+D3 concern small-scale systems (i.e., the number of entities is small), so we only use them in Section III-D2 to show that model sharing is also feasible in small-scale systems. The detailed statistics of all these datasets are shown in Table I.

As discussed in Section III-A1, the entity patterns are diverse but repeatable in an online system. Taking the entities in TC_data as examples, the visualization of data of some entities are displayed in Fig. 2. We can observe that there exist diverse entity patterns (e.g., "IP 192", "IP 459" and "IP 526") in an online system, and meanwhile, there also exist entities with similar patterns (e.g., "IP 459" and "IP 472"). Similar phenomenons occur in SMD, JS_D2+D3 and CTF_data too. Readers can visualize these datasets for more details. Moreover, we will further analyse this phenomenon quantitatively in Section III-D1.

Each dataset will go through a preprocessing process before fed into the models. For the public datasets (i.e., SMD, JS_D2+D3 and CTF_data), we utilize the preprocessing method recommended in their original papers. Specifically, SMD is preprocessed by the z-score normalization, JS_D2+D3 is preprocessed by the min-max normalization. CTF_data and TC_data utilize the iterative z-score normalization proposed in CTF [17].

The experiments are conducted on a server with an Intel(R) Xeon(R) Gold 6242 CPU, 4 NVIDIA Tesla V100 GPUs, and a 125GB RAM.

D. RQ1: Study on the Feasibility of Model Sharing

1) RQ1-1: Understanding and Identifying the Pattern Repeatability in an Online System: To quantitatively evaluate the pattern repeatability, we begin with some concepts, including the period-wise similarity and the entity-wise repeatability. These concepts can help illustrate why trained models can work for unsupervised anomaly detection, and further, why they might still work under a sharing setting.

TABLE II: The average period-wise similarity and the entity-wise repeatability in two representative datasets

Datasets	φ_{period}	ϱ_{entity}
CTF_data	0.40	13%
TC_data	0.36	8%

Period-Wise Similarity. Typically, metrics from modern online systems exhibit periodicity (e.g., in days). This property is the key which supports the working of deep learning based anomaly detectors. Specifically, it is the repeatable pattern in metric data along time that allows deep models trained on the historical data to work on upcoming observations. In this study, we define a measure, namely the period-wise similarity, to quantitatively evaluate the pattern repeatability in metric data along time. Specifically, the period-wise similarity $\varphi_{period}(e)$ of a monitored entity e is the average similarity between all pairs of its historical cycles, i.e.,

$$\varphi_{period}(e) = \frac{\sum_{i \neq j \wedge i,j \in \mathcal{T}} \rho(\mathbf{X}_{i:i+T}(e), \mathbf{X}_{j:j+T}(e))}{P^2_{|\mathcal{T}|}}, \quad (1)$$

where $\mathcal{T}$ denotes a set containing the cycle starting time points (e.g., 00:00 at each day), T refers to the length of a cycle (e.g., one day), $\mathbf{X}_{i:i+T}(e)$ and $\mathbf{X}_{j:j+T}(e)$ represent some cycles in metric data of the monitored entity e, and $P^2_{|\mathcal{T}|}$ is the number of historical cycle pairs, calculated as the number of permutations of 2 elements selected from the set $\mathcal{T}$. With respect to the operation $\rho(\cdot)$ in the equation, we select the Pearson Correlation Coefficient to calculate the overall similarity between two cycles.

Entity-Wise Repeatability. The pattern repeatability along time can account for why a deep learning based anomaly detection model works on upcoming observations after trained on historical data, and the period-wise similarity (i.e., Equation (1)) exposes the strength of such pattern repeatability. To further gain an insight into why model sharing may work, we shall dive into the repeatable patterns among different monitored entities, namely the entity-wise repeatability. To evaluate the entity-wise repeatability, we need to first construct a representation for each monitored entity which can well

TABLE III: The efficacy comparison of some representative deep anomaly detection models under different settings

Datasets	Models	point-adjusted F1-score *one-for-one* $\rightarrow$ *model-sharing*
CTF_data	AE [27]	**0.440** $\rightarrow$ **0.730 (+0.290)** $\uparrow$
	LSTMED [28]	**0.619** $\rightarrow$ **0.766 (+0.147)** $\uparrow$
	Omni [12]	**0.678** $\rightarrow$ **0.690 (+0.012)** $\uparrow$
	USAD [14]	**0.530** $\rightarrow$ **0.598 (+0.068)** $\uparrow$
	SDFVAE [16]	**0.651** $\rightarrow$ **0.753 (+0.102)** $\uparrow$
TC_data	AE [27]	**0.714** $\rightarrow$ **0.824 (+0.110)** $\uparrow$
	LSTMED [28]	**0.745** $\rightarrow$ **0.780 (+0.035)** $\uparrow$
	Omni [12]	**0.674** $\rightarrow$ **0.811 (+0.137)** $\uparrow$
	USAD [14]	**0.683** $\rightarrow$ **0.792 (+0.109)** $\uparrow$
	SDFVAE [16]	**0.745** $\rightarrow$ **0.817 (+0.072)** $\uparrow$
JS_D2+D3	AE [27]	**0.931** $\rightarrow$ **0.947 (+0.016)** $\uparrow$
	LSTMED [28]	**0.967** $\rightarrow$ **0.952 (-0.015)**
	Omni [12]	**0.950** $\rightarrow$ **0.965 (+0.015)** $\uparrow$
	USAD [14]	**0.890** $\rightarrow$ **0.920 (+0.030)** $\uparrow$
	SDFVAE [16]	**0.959** $\rightarrow$ **0.947 (-0.012)**
SMD	AE [27]	**0.891** $\rightarrow$ **0.913 (+0.022)** $\uparrow$
	LSTMED [28]	**0.913** $\rightarrow$ **0.922 (+0.009)** $\uparrow$
	Omni [12]	**0.875** $\rightarrow$ **0.912 (+0.037)** $\uparrow$
	USAD [14]	**0.903** $\rightarrow$ **0.903 (+0.000)**
	SDFVAE [16]	**0.924** $\rightarrow$ **0.932 (+0.008)** $\uparrow$

describe its regular pattern. Towards this goal, we introduce a concept, namely the **representative curve**. The representative curve $\widehat{\mathbf{X}}(e) \in \mathbb{R}^{N\times T}$ of a monitored entity can be calculated using its historical metric data, as follows:

$$\widehat{\mathbf{X}}(e) = \text{median}_{i\in\mathcal{T}}(\mathbf{X}_{i:i+T}(e)), \tag{2}$$

where $\text{median}(\cdot)$ denotes a point-wise median operation over all cycles in the historical metric data. Through this median operation, we can remove noises or some transient patterns in historical data and create a representation which is representative enough for the monitored entity. Then, based upon the acquired representative curves, we can estimate the entity-wise similarity $\varphi_{entity}(e, u)$ between two monitored entities (e.g., e and u), as follows,

$$\varphi_{entity}(e, u) = \rho(\widehat{\mathbf{X}}(e), \widehat{\mathbf{X}}(u)), \tag{3}$$

where the operation $\rho(\cdot)$ is the same as that used in Equation (1). In this study, the patterns of two entities are considered to be similar if their entity-wise similarity is over the average period-wise similarity of all entities from the system. Then, the entity-wise repeatability ϱ_{entity} of a system is formulated as the proportion of similar entity pairs in the system.

Table II presents the average period-wise similarity and the entity-wise repeatability of two representative datasets. These two datasets are selected here because they provide timestamp information for the calculation of the representative curves(i.e., Equation (2)). Besides, they are the most large-scale datasets for anomaly detection on metric data, which are exactly what we target at. It can be observed that there exists a desirable level of pattern repeatability among entities in large-scale online systems. Such repeatability can ease the burden of the learning of anomaly detection models on abundant entities. Moreover, such repeatability can allow model sharing act as a type of data augmentation [24], and ultimately boost the efficacy of the shared model.

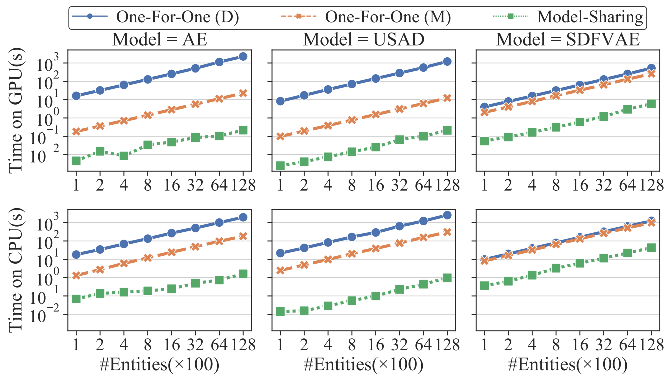


Fig. 3: The inference speed of some representative deep unsupervised anomaly detection models

2) RQ1-2: An Experimental Study on the Efficacy of Some Representative Deep Anomaly Detection Models under Different Settings: In this section, we will validate that deep models can model diverse patterns across different entities and still maintain their anomaly detection effect when the model is shared. In particular, we will examine the efficacy of some basic or state-of-the-art deep anomaly detection models, including AE [27], LSTMED [28], Omni [12], USAD [14], SDFVAE [16], under different settings. The best point-adjusted F1-score, which is widely used in previous literature [12], [14], [16], [18], is selected as an indicator of their efficacy here (more details are stated in Section V-C). Table III shows the experimental results on two large-scale datasets and two small-scale datasets. We can see that all trials gain nearly unchanged or improved results when the model usage setting is switched from a one-for-one setting to a model-sharing setting. Overall, these results lead us to conclude with minor exceptions that in large-scale online systems, deep anomaly detection models can be still effective under a model-sharing setting.

E. RQ2: Study on the Necessity of Model Sharing

To shed light on how model sharing affects the model inference efficiency in practice, we conduct an experimental study using some representative deep unsupervised anomaly detection models. In particular, we sample a bundle of data items from CTF_data to simulate metric data collected from various monitored entities, and then evaluate the execution time needed for model inference under different settings, including: (i) **One-For-One (D)**: A one-for-one setting is adopted, and the corresponding model for each entity needs to be reloaded from the disk one by one during model inference. (ii) **One-For-One (M)**: A one-for-one setting is adopted, and all models for different entities can reside in the memory. (iii) **Model-Sharing**: A model-sharing setting is adopted.

Fig. 3 shows the experimental results on GPU and CPU using three representative models (AE [27], USAD [14] and SDFVAE [16]), respectively. We find that when a model-sharing setting is adopted, the evaluated representative models can complete the inference within a satisfactory time, even when the number of entities is enormous. For example, in our experiments, when the number of entities reaches 12800, the inference time on GPU for AE, USAD and SDFVAE are only 0.22s, 0.21s and 5.95s, respectively, and those on CPU are 1.61s, 0.98s and 43.04s, respectively. However, when a one-for-one setting is adopted, the inference time is far longer and unsatisfactory, nearly with a magnification of 100x, or even a magnification of 10000x when the models have a large size and need to be reloaded from the disk. Under this circumstance, the inference time not only depends on the model efficiency, but also highly depends on the model size. This is because when the model size is large, it is more difficult to preserve all models in the memory under a one-for-one setting. At the same time, the model loading time of each model will be longer. It is quite common because deep models are usually heavy-weighted. For example, the sizes of AE, USAD, SDFVAE are 244MB, 177MB and 6MB, respectively, when trained on CTF_data with the default hyper-parameters. Consequently, unless a labor-intensive effort is paid to hyper-parameter tuning until a desired balance between efficacy and efficiency is reached, or an excessive amount of computational resources are committed to parallelize the inference process, such a long inference time brought by a one-for-one setting is unsatisfactory in practice.

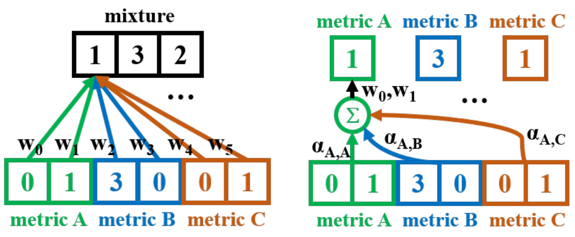
IV. Proposed Model

A. Challenges for Model Sharing

Though we have shown that model sharing is feasible, directly applying previous models in a model-sharing setting can still face some additional challenges.

Challenge 1: Though the entity behavior patterns are repeatable in an online system, they are still not stable (e.g., from Fig. 2(d), we can see that some entities exhibit different behavior patterns). If the deep model is not designed to consider the diverse entity patterns, the anomaly detection efficacy can still be low. To sum up, the first challenge for model sharing is, how to design a deep anomaly detection model that can learn various entity behavior patterns in a large-scale online system?

Challenge 2: Since the model needs to be shared across different entities, it may come across variable-length inputs. Variable-length inputs denote data with variable number of features (e.g., different monitored entities with variable number of metrics). In fact, variable-length inputs are quite common in online system management, and it might hinder a successful adoption of model sharing to some extent. Inputs of this type can be generally classified into three categories according to the causes. The first category is caused by the existence of heterogeneous entities (e.g., a service and a machine) in modern online systems. The second category stands for variable-length inputs caused by the existence of some long-term missing data (missed for a couple of hours or even days). The third category refers to variable-length inputs caused by the need for masking some noisy metrics in some specific monitored entities. In particular, there exist some metrics which are helpful in only a portion of monitored entities, but are too noisy and may cause a lot of false alarms in other entities. Therefore, on-call



(a) Binding based components. (b) Position based components.

Fig. 4: Examples of binding based components and position based components with an input of size N=3, W=2

engineers may need to mask such metrics in some entities to wipe out their negative effect, which generates variable-length inputs. In a word, models that are capable of handling variable-length inputs are more preferred for model sharing. Then, the second challenge for model sharing is, how to design a deep anomaly detection model that can accept and process variable-length inputs.

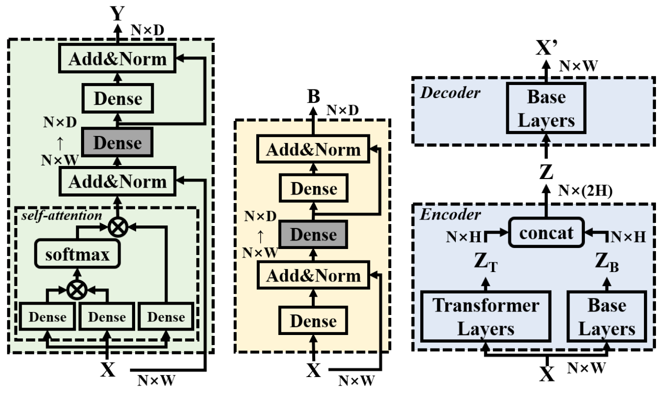
B. Network Architecture

Model Design Motivation. To show why conventional deep models cannot cope with the aforementioned challenges well, in Fig. 4(a), we visualize the neural network components that they usually use to construct the model. For simplicity, we name components of this type as binding based components. Binding based components process multivariate inputs by binding each dimension to each specific weight. It is simple and straightforward. Therefore, it has been widely adopted by many previous deep unsupervised models [12], [14], [16], [27], [28]. However, in a binding based component, weights are individually assigned to each dimension (e.g., w_0, w_1, etc. in Fig. 4(a)) and need to be fixed for all inputs from all entities after trained. When the trained model needs to be shared across entities that contain diverse patterns, weight assignment in such a rigid way can penalize a proper modelling. This is because the same dimension in different entities might exhibit different behaviors, rendering the binding between a specific dimension and a corresponding weight inappropriate. Besides, binding based components are not extensible for variable-length inputs. Therefore, we do not use binding based components to construct our model. Instead, we carefully organize the inputs and construct a model that only contains position-based components (visualized in Fig. 4(b)) to process them. In a position based component, the corresponding weights for each dimension (e.g., $\alpha_{A,A}$, $\alpha_{A,B}$, etc. in Fig. 4(b)) can be updated according to the input using techniques such as self-attention mechanism, and some shared weights (e.g., w_0, w_1, etc. in Fig. 4(b)) are held for the learning of common patterns across dimensions. This is a more appropriate way when the model needs to be shared across entities with diverse patterns. Moreover, with a careful input organization, we can use position based components to process variable-length inputs, which is depicted below.

Model Overview. Fig. 5 shows an overview of Uni-AD. Uni-AD models the metric patterns by reconstructing them through an auto-encoder based architecture as shown in Fig. 5(c) and detects anomalies using the reconstruction error. The calculations are performed by two basic building blocks, namely the Transformer encoder layer (Fig. 5(a)) and the Base layer (Fig. 5(b)). Both of these two building blocks are position-based components. Specifically, unlike previous transformer based time series analysis methods [29], [30], which treat each timestamp as a position, Uni-AD treats each metric as a position to allow the model extensible for variable-length inputs.

Transformer Encoder Layers [31] are mainly composed of the self-attention mechanism and position-wise fully connected networks (also denoted as dense layers in the following). Meanwhile, it is equipped with various advanced deep learning techniques such as residual connections [32] and layer normalization [33]. A detailed description about its forwarding can be referred to in the original work [31]. Notably, to help perform dimension reduction in Uni-AD with the Transformer encoder layer, we add a dense layer between its self-attention mechanism and position-wise fully connected networks. A diagram of the modified version is presented in Fig. 5(a). In particular, the Transformer encoder layer takes an observation $\mathbf{X} \in \mathbb{R}^{N \times W}$ as input. Then it treats each of its metrics (i.e., a sliding window $\mathbf{X}_i \in \mathbb{R}^{W}$, where i denotes one of the metrics) as a position, and calculates their middle representations $\mathbf{Y}_i \in \mathbb{R}^{D}$ (D is the middle representation size) accordingly with the consideration of inter-position patterns using the self-attention mechanism. Middle representations of all the metrics constitute the outputted middle representation $\mathbf{Y} \in \mathbb{R}^{N \times D}$. Through stacking multiple Transformer encoder layers, a part of the final hidden representation $\mathbf{Z}_T \in \mathbb{R}^{N \times H}$ (H is the hidden representation size) can be calculated. With its help, the inter-metric patterns can be well embedded in the hidden representations. The modelling of the inter-metric patterns can aid the mitigation of the Challenge 1. Moreover, we omit the positional encoder part in the original Transformer design [31]. This is because including the positional encoder will make Uni-AD exhibit some drawbacks of the binding-based components. Therefore, if we change the order of the metrics of an entity when they are fed into Uni-AD, the detection result will be the same as that of the original order. This is reasonable because anomalies are mainly reflected in the metric pattern, while not the order. With this design, Uni-AD can naturally accept and process variable-length inputs without some other specifications (e.g., the order).

Base Layers are building blocks designed in this study to remedy the weakness of simply stacking multiple Transformer encoder layers. Specifically, though the self-attention mechanism in a Transformer encoder layer enables the learning of inter-metric patterns, it may cause an over-smoothing representation [34]–[36] due to its dimension-fusing nature (i.e., "$\sum$" in Fig. 4(b)). If the hidden representations of different metrics converge to a similar value, their expressive power will degrade and can hinder a successful modelling under a



(a) Transformer encoder layer (b) Base layer (c) The overall network architecture

Fig. 5: An overview of the proposed model

model-sharing setting. Therefore, as shown in Figure 5(b), we design a new layer as the complement of Transformer encoder layer, namely the Base layer, by substituting the self-attention mechanism in the Transformer encoder layer with a simple dense layer. This design enables it to concentrate on the compression and reconstruction based on the information from the metrics themselves. Taking an observation $\mathbf{X} \in \mathbb{R}^{N \times W}$ as input, the Base layer outputs its middle representation $\mathbf{B} \in \mathbb{R}^{N \times D}$ through incorporating multiple dense layers with layer normalization and residual connections. A part of the final hidden representation $\mathbf{Z}_B \in \mathbb{R}^{N \times H}$ is calculated through stacking multiple Base layers. Besides, the decoder of Uni-AD is constructed using the Base layers.

Overall, using the Transformer Encoder layers and the Base layers to construct the encoder, and the Base layers as the decoder, the feed forward mechanism of Uni-AD is as follows,

$$\begin{aligned}
\mathbf{Z}_T &= Transformer\_Encoder(\mathbf{X}) \\
\mathbf{Z}_B &= Base\_Encoder(\mathbf{X}) \\
\mathbf{Z} &= concat(\mathbf{Z}_T, \mathbf{Z}_B) \\
\mathbf{X}' &= Base\_Decoder(\mathbf{Z}).
\end{aligned} \tag{4}$$

Here, $concat(\cdot)$ is a concatenation operation, which merges the information in two hidden representations $\mathbf{Z}_T$ and $\mathbf{Z}_B$, and calculate the final hidden representation $\mathbf{Z} \in \mathbb{R}^{N \times 2H}$. Then, $\mathbf{X}' \in \mathbb{R}^{N \times W}$ is the reconstruction calculated by the decoder of Uni-AD using $\mathbf{Z}$.

C. Model Training

Uni-AD can be trained by optimizing the reconstruction error of the inputs. There exist many distance measures which can represent the reconstruction error. In this study, we use the Huber loss [37] as the default training objective of Uni-AD. It is less sensitive to outliers in data than the mean square error (mse) which is widely used previously. The Huber loss can be formalized as:

$$\mathcal{L}(\mathbf{X}_{i,j}, \mathbf{X}'_{i,j}) = \begin{cases} 0.5(\mathbf{X}_{i,j} - \mathbf{X}'_{i,j})^2 & \text{if } |\mathbf{X}_{i,j} - \mathbf{X}'_{i,j}| < 1 \\ |\mathbf{X}_{i,j} - \mathbf{X}'_{i,j}| - 0.5 & \text{otherwise} \end{cases}, \tag{5}$$

$$\mathcal{L}(\mathbf{X}, \mathbf{X}') = \frac{1}{N * W} \sum_{i=1}^{N} \sum_{j=1}^{W} \mathcal{L}(\mathbf{X}_{i,j}, \mathbf{X}'_{i,j}). \tag{6}$$

Here, $\mathbf{X}'$ denotes the reconstruction of the input $\mathbf{X}$, and $\mathbf{X}'_{i,j}$ is the i^{th} metric at the j^{th} timestamp in $\mathbf{X}'$.

D. Model Inference

During online inference, operators usually care about whether an anomaly is occurring at the current time (i.e., timestamp t) and pursue a timely detection result, so we apply the reconstruction error of the inputted observation at timestamp t as its anomaly score s_t to indicate whether an anomaly is occurring or not, i.e.,

$$s_t = \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} (\mathbf{X}_{i,t} - \mathbf{X}'_{i,t})^2, \tag{7}$$

where $\mathcal{M}$ is a selected set of metrics by the operators (default to be the whole metric set). Note that when $|\mathcal{M}| < N$, the corresponding dimensions of $\mathbf{X}$ are removed before fed into the model. When s_t is greater than a predefined threshold τ, which can be selected by a 3-σ rule [38], or other methods [12], [39], an alert will be triggered.

V. Evaluation of the Proposed Model

A. Research Questions

In this section, we conduct experiments to answer the following research questions about the proposed model:

- **RQ3. Comparisons:** Can Uni-AD outperform other state-of-the-art models in unsupervised anomaly detection for metric data under a model-sharing setting?
- **RQ4. Ablation Study:** How is the effectiveness of each design choice in Uni-AD?
- **RQ5. Benefits from Handling Variable-length Inputs:** How can the ability of Uni-AD to handle variable-length inputs benefit the anomaly detection performance?
- **RQ6. Efficiency:** How is the efficiency of Uni-AD?

B. Implementation

We implement Uni-AD based on PyTorch [25]. In Uni-AD, the number of layers (Transformer encoder layers and Base layers) in encoder is set as 2, and the number of Base layers in decoder is set as 2 too. By default, each layer in encoder will calculate an output whose size equals a power of 2 and is greater than and nearest to half of the input size, and each layer in decoder will calculate an output whose size equals a multiple of the hidden representation size and is nearest to the corresponding encoder layer size calculated above. We use the Adam optimizer to train our model. The number of maximum training epochs is set as 5 and the batch size is set as 128. During training, 25% of data in the training dataset will be used as the validation set and an early stop strategy is applied according to the validation loss at the end of each epoch.

C. Evaluation Measures

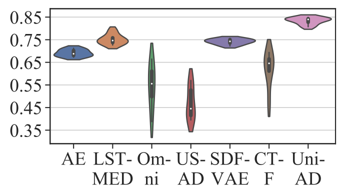
We validate our model by F1-score and Area Under Curve (AUC). F1-score is the harmonic mean of Precision and Recall. Following previous work [7], [12], [14], [16], [18], we enumerate thresholds and use the best F1-score as the final result for it. As for AUC, Area Under Precision-Recall Curve (PR-AUC) is selected since it is reported to be more suitable for high-skewed datasets compared with Area Under ROC Curve (ROC-AUC) [40]. Among these two metrics, the best F1-score evaluates the result at the best of times, exposing the potential of a model, while PR-AUC gives an overall and more comprehensive evaluation of the model.

Since classical point-based Precision and Recall are inadequate to represent domain-specific performance for anomaly detection on time series data (e.g., metric data in this study), two strategies, namely the point-adjusted strategy [12] and the range-based strategy [41], have been proposed. For the point-adjusted strategy, if any point in a ground-truth anomaly segment is correctly identified as an anomaly by a model, it will consider all other points in the segment as true positives when evaluating Precision and Recall. This strategy has been widely utilized in previous work [12], [14], [16], [18]. By comparison, the range-based strategy is stricter and considers more aspects of the predictions, including the existence, size, position and cardinality. More details can be found in the original work [41]. In this study, we consider both of these two strategies to give a comprehensive evaluation of the proposed model.

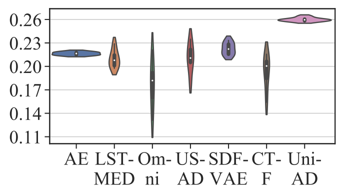
D. RQ3: Comparisons

To demonstrate the effectiveness of Uni-AD, we compare it with 2 basic models, including AE [27], LSTMED [28], and 3 up-to-date models which are applied in the background of online system maintenance, including Omni [12], USAD [14] and SDFVAE [16], as well as a clustering based framework for scalable anomaly detection, CTF [17]. Unlike previous studies which only select and report the best result from several trials of different hyper-parameters, we report all the results of these models with hyper-parameters set as several frequent-used or default values. This is because the result produced with the best hyper-parameters is actually selected based upon test data labels, which exactly violates the assumption of an unsupervised setting, and thus probably unachievable and even meaningless in practice. Therefore, we focus on the comparison about the overall performance of these models in this study. Specifically, we select some common hyper-parameters (i.e., the hidden representation size and the learning rate) and enumerate them as different regular-used values (i.e., 3, 8 and 16 for the hidden representation size and 1E-3 and 1E-4 for the learning rate) to repeat the experiments.

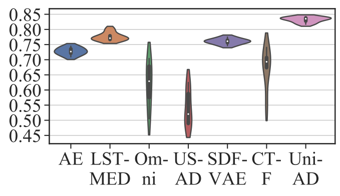
Fig. 6 and 7 present the experimental results on CTF_data and TC_data in the form of violin charts. Overall, Uni-AD achieves state-of-the-art results over all horizons on both of



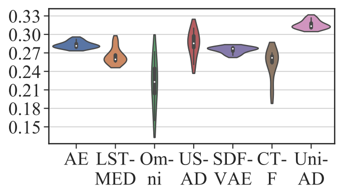
(a) Point-adjusted PR-AUC



(b) Range-based PR-AUC

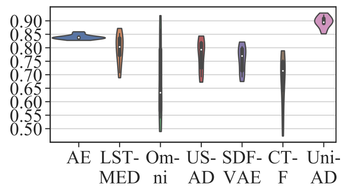


(c) Best point-adjusted F1-score

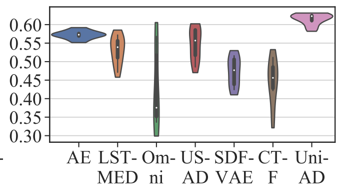


(d) Best range-based F1-score

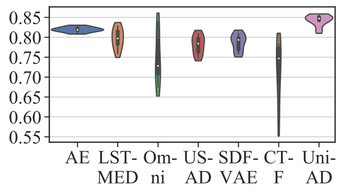
Fig. 6: Comparisons on CTF_data



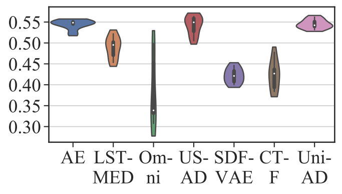
(a) Point-adjusted PR-AUC



(b) Range-based PR-AUC

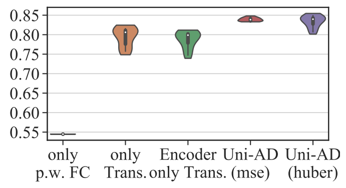


(c) Best point-adjusted F1-score

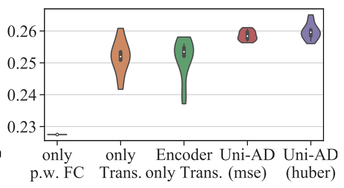


(d) Best range-based F1-score

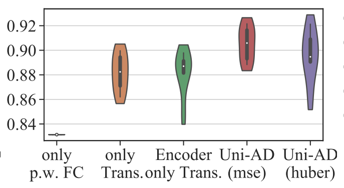
Fig. 7: Comparisons on TC_data



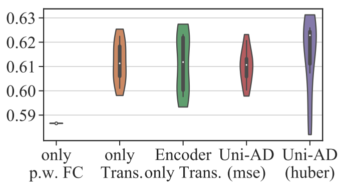
(a) Point-adjusted PR-AUC on CTF_data



(b) Range-based PR-AUC on CTF_data



(c) Point-adjusted PR-AUC on TC_data



(d) Range-based PR-AUC on TC_data

Fig. 8: Ablation Studies on CTF_data and TC_data

the datasets. Specifically, in point-adjusted PR-AUC, compared with all baselines, Uni-AD achieves relative improvements of 11.01%$\sim$77.71% on CTF_data and 6.66%$\sim$32.17% on TC_data, and in range-based PR-AUC, its average relative improvements are 16.72%$\sim$ 46.49% on CTF_data and 7.49%$\sim$43.97% on TC_data. Besides, the performance of Uni-AD is more stable than a couple of baselines.

The inferior performance of conventional models (i.e., AE, LSTM-ED, Omni, USAD and SDFVAE) is in part caused by their binding based components, which hinder a proper modelling of diverse patterns in data when the model is shared. Meanwhile, we can observe that the performance of basic models (i.e., AE, LSTMED) is not doom to behave worse than up-to-date models and frameworks (i.e., Omni, USAD, SDFVAE, CTF) under a model-sharing setting. This observation implies that the efficacy of advanced designs proposed in previous studies needs to be reexamined if a model-sharing setting is adopted for practicality.

CTF is proposed for scalable anomaly detection, but in our experiments, it presents an unstable and suboptimal performance. This is in part caused by the sub-sampling mechanism that it utilizes to accelerate the training. The performance of CTF heavily relies on the quality of its sub-sampled data for training, while for these two datasets, there is not enough information provided for a decision about which training data should be kept. As a result, only a random sub-sampling strategy can be utilized and thus CTF could not consistently achieve a satisfactory result.

E. RQ4: Ablation Study

To validate each design choice of Uni-AD, we conduct ablation studies using variants of Uni-AD, including: (i) **Only p.w. FC:** Substituting all Transformer encoder layers and Base layers of Uni-AD into position-wise Fully Connected networks, the most basic position-based component. (ii) **Only Trans.:** Substituting all Base layers of Uni-AD into Transformer encoder layers. (iii) **Encoder only Trans.:** Removing the Base layers in the encoder of Uni-AD. (iv) **Uni-AD (mse):** Uni-AD trained with the mean square error loss.

As in Fig. 8, Uni-AD consistently outperforms or matches all its variants. Among the variants, "Only p.w. FC" is composed of only the most basic position based components (i.e., the component denoted in Fig. 4(b) when $\alpha_{i,j} = 1$ if $i = j$ and 0 otherwise). Though its performance is rather stable under different hyper-parameters, it cannot compete with Uni-AD and other variants. This demonstrates that the advanced techniques included in Transformer encoder layers and Base layers play an important role in a proper modelling

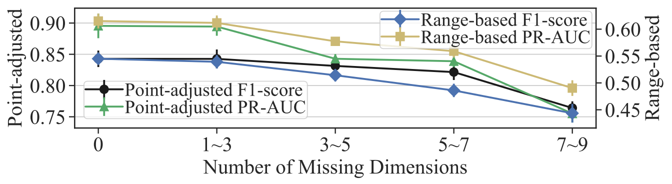


Fig. 9: The efficacy of Uni-AD on TC_data with different numbers of missing dimensions

of diverse patterns, which contribute to the anomaly detection performance. As for the comparisons with "Only Trans." and "Encoder only Trans.", the superiority of Uni-AD indicates the effectiveness of Base layers for ensuring the expressivity of calculated representations and improving the anomaly detection performance.

With respect to the selection of different loss functions, "Uni-AD (huber)" and "Uni-AD (mse)" behave similarly. However, we still prefer the Huber loss as the default choice of loss function in Uni-AD because it has a potential to prevent over-fitting to outliers in training data as declared in previous literature [37]. Meanwhile, utilizing the Huber loss would not bring too much extra overhead.

F. RQ5: Handling Variable-length Inputs

As introduced in Section IV-A, variable-length inputs include 3 categories, namely inputs from heterogeneous entities, inputs with long-term missing dimensions and inputs with noisy dimensions masked. Since inputs from heterogeneous entities can be solved by training a corresponding model for each type of entities, which is feasible because the number of entity types is far less than the number of entities, we mainly investigate the efficacy of Uni-AD on the latter two types of variable-length inputs.

1) Handling Long-term Missing Metrics: To validate the ability of Uni-AD in handling data with long-term missing metrics, we simulate this type of input using TC_data by dropping random metrics of each entity when testing the model. Fig. 9 shows the efficacy of Uni-AD on data under different missing rates. Specifically, the x-axis denotes the number of simulated missing metrics (e.g., "1∼3" means that for each entity, a quantity in a range from 1 to 3 of metrics will be randomly selected as long-term missing metrics and removed from the inputs). We can observe that Uni-AD can retain a stable performance and remains its superiority over the baseline models (according to their results on the complete data as in Fig. 7) when a few metrics (e.g., $<=$ 3 metrics) are missing. For the baseline models, these inputs cannot even be handled directly unless a new customized model is trained for them. Only when over a half of metrics are missing (i.e., $>$ 5 metrics for TC_data) will the efficacy of Uni-AD be seriously affected. Overall, the results demonstrate the considerable ability of Uni-AD in handling data containing long-term missing metrics.

2) Masking Noisy Metrics: As discussed in Section IV-A, noisy metrics in some specific monitored entities may have a negative impact on the anomaly detection performance. If operators notice such metrics and want to wipe out their negative impact through some configurations, Uni-AD can help because it can process inputs with metrics masked. To show this, we select three noisy monitored entities from TC_data according to Signal-to-Noise Ratio [42] and the judgement of operators. For these entities, if we mask the noisy metrics, Uni-AD can gain an average improvement of 0.48, 0.32, 0.18, and 0.26 in point-adjusted PR-AUC, range-based PR-AUC, point-adjusted F1-score and range-based F1-score, respectively. In a word, the ability of Uni-AD in handling inputs with metrics masked is useful. With this ability, the anomaly detection performance can be further improved in some cases.

TABLE IV: The time and space efficiency of Uni-AD

Device	Training time (min)	Detection time per sample (sec)	Model size (KB)
GPU	20.47∼21.25	5.26E-5∼6.32E-5	355.11∼488.93
CPU	82.16∼96.88	4.34E-4∼4.88E-4	

G. RQ6: Efficiency

In this section, we study the efficiency of Uni-AD in terms of training time, detection time and model size on TC_data with the hyper-parameters used in Section V-D. As in Table IV, training Uni-AD on a total of 500 entities needs only around 20 minutes on GPU and around 90 minutes on CPU, and the per-sample detection time is less than 100μs on GPU and less than 1ms on CPU. Since this base value is small and the training and detection time is linearly correlated with the number of entities, the training and detection time will be still acceptable for a more large-scale system (e.g., with thousands of entities). To sum up, Uni-AD is efficient enough for a practical deployment in real-world modern online systems.

H. Discussion

Limitation. This study mainly concerns the availability monitoring for modern online systems. In some other anomaly detection scenarios, such as intrusion detection in some safety-critical applications, applying model sharing needs to be more careful. For example, engineers may need to carefully audit the shared training data to avoid backdoor attacks [43]. In these scenarios, customized anomaly detection methods discussed in Section I is more appropriate currently. In the future, we will consider constraints in different scenarios and investigate how to extend model sharing to more scenarios.

VI. Related Work

Deep Unsupervised Anomaly Detection Models. Recently, deep learning attracts tremendous attention due to their expressive power for complex data. Under this background, many deep models are proposed for unsupervised anomaly detection in modern online systems [7]–[17], [19], [44]. For instance, Omni [12] combines stochastic recurrent neural networks, stochastic variable connections and planar normalizing flow to perform robust anomaly detection. USAD [14] performs anomaly detection based on auto-encoders trained with adversarial training techniques. SDFVAE [16] applies a VAE based architecture, and it further explicitly factorizes the hidden representations into static and dynamic parts to construct a noise-resistant model. However, all of these models are proposed and evaluated under a one-for-one setting, and thus they might lose the practicality in a modern online system. There also exists work which tries to improve the scalability of anomaly detection models, such as ROCKA [20] and CTF [17]. Both of them use a clustering based method to help achieve scalable anomaly detection, while ROCKA focuses on anomaly detection for univariate time series and CTF focuses on that for multivariate time series.

Expressivity and generalization of Deep Models. Some previous work discusses the expressivity and generalization of deep models, which provide some supporting theories for why model sharing works. For example, Zhang, et al. [22] define a notion of effective capacity, and find that deep neural networks have a powerful fitting capacity and can even easily fit random labels. Yosinski, et al. [23] study the transferabilty of deep neural networks. Battaglia et al. [45] discuss inductive biases of several deep models, which can inspire some design choices in Uni-AD.

VII. Conclusion

In this paper, we study on the question "Should we share a unified model when utilizing deep models for unsupervised anomaly detection in modern online systems?". After comprehensive evaluations and comparisons, the answer is "yes" for an online system where entity patterns are repeatable. Based on a dozen of experimental studies, we demonstrate the feasibility and necessity of the model-sharing setting for deep unsupervised anomaly detection in modern online systems. Besides, we propose a novel model Uni-AD, which overcomes some drawbacks of previous models under a model-sharing setting. Extensive experiments on two large-scale datasets collected from real-world systems demonstrate the superiority of Uni-AD over other state-of-the-art baselines.

Acknowledgments

The research is supported by the National Key Research and Development Program of China (2019YFB1804002), the National Natural Science Foundation of China (No. U1811462), the Basic and Applied Basic Research of Guangzhou (No. 202002030328), the Natural Science Foundation of Guangdong Province (No.2019A1515012229), and Tencent Rhino-Bird Joint Research Program (No.K21-67000-059). The corresponding author is Pengfei Chen.

References

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

[2] N. Dragoni, S. Giallorenzo, A. Lluch-Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: Yesterday, today, and tomorrow," in *Present and Ulterior Software Engineering*, M. Mazzara and B. Meyer, Eds. Springer, 2017, pp. 195–216.

[3] H. Zhou, M. Chen, Q. Lin, Y. Wang, X. She, S. Liu, R. Gu, B. C. Ooi, and J. Yang, "Overload control for scaling wechat microservices," in *Proceedings of the ACM Symposium on Cloud Computing, SoCC 2018, Carlsbad, CA, USA, October 11-13, 2018.* ACM, 2018, pp. 149–161.

[4] M. Ma, Z. Yin, S. Zhang, S. Wang, C. Zheng, X. Jiang, H. Hu, C. Luo, Y. Li, N. Qiu, F. Li, C. Chen, and D. Pei, "Diagnosing root causes of intermittent slow queries in large-scale cloud databases," *Proc. VLDB Endow.*, vol. 13, no. 8, pp. 1176–1189, 2020.

[5] R. Tang, Z. Yang, Z. Li, W. Meng, H. Wang, Q. Li, Y. Sun, D. Pei, T. Wei, Y. Xu, and Y. Liu, "Zerowall: Detecting zero-day web attacks through encoder-decoder recurrent neural networks," in *39th IEEE Conference on Computer Communications, INFOCOM 2020, Toronto, ON, Canada, July 6-9, 2020.* IEEE, 2020, pp. 2479–2488.

[6] A. Siffer, P. Fouque, A. Termier, and C. Largouët, "Anomaly detection in streams with extreme value theory," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017.* ACM, 2017, pp. 1067–1075.

[7] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018.* ACM, 2018, pp. 187–196.

[8] Z. Li, W. Chen, and D. Pei, "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, 2018, pp. 1–9.

[9] W. Chen, H. Xu, Z. Li, D. Pei, J. Chen, H. Qiao, Y. Feng, and Z. Wang, "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, 2019, pp. 1891–1899.

[10] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S. Ng, "MAD-GAN: multivariate anomaly detection for time series data with generative adversarial networks," in *Artificial Neural Networks and Machine Learning - ICANN 2019: Text and Time Series - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part IV*, ser. Lecture Notes in Computer Science, vol. 11730. Springer, 2019, pp. 703–716.

[11] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019.* AAAI Press, 2019, pp. 1409–1416.

[12] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* ACM, 2019, pp. 2828–2837.

[13] L. Shen, Z. Li, and J. T. Kwok, "Timeseries anomaly detection using temporal hierarchical one-class network," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[14] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "USAD: unsupervised anomaly detection on multivariate time series," in *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020.* ACM, 2020, pp. 3395–3404.

[15] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang, "Multivariate time-series anomaly detection via graph attention network," in *20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020.* IEEE, 2020, pp. 841–850.

[16] L. Dai, T. Lin, C. Liu, B. Jiang, Y. Liu, Z. Xu, and Z. Zhang, "SDFVAE: static and dynamic factorized VAE for anomaly detection of multivariate CDN kpis," in *WWW '21: The Web Conference 2021, Virtual Event /*

Ljubljana, Slovenia, April 19-23, 2021. ACM / IW3C2, 2021, pp. 3076–3086.

[17] M. Sun, Y. Su, S. Zhang, Y. Cao, Y. Liu, D. Pei, W. Wu, Y. Zhang, X. Liu, and J. Tang, "CTF: anomaly detection in high-dimensional time series with coarse-to-fine model transfer," in *40th IEEE Conference on Computer Communications, INFOCOM 2021, Vancouver, BC, Canada, May 10-13, 2021*. IEEE, 2021, pp. 1–10.

[18] M. Ma, S. Zhang, J. Chen, J. Xu, H. Li, Y. Lin, X. Nie, B. Zhou, Y. Wang, and D. Pei, "Jump-starting multivariate time series anomaly detection for online service systems," in *2021 {USENIX} Annual Technical Conference ({USENIX} {ATC} 21)*, 2021, pp. 413–426.

[19] A. Abdulaal and et.al., "Practical approach to asynchronous multivariate time series anomaly detection and localization," in *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, F. Zhu, B. C. Ooi, and C. Miao, Eds. ACM, 2021, pp. 2485–2494.

[20] Z. Li, Y. Zhao, R. Liu, and D. Pei, "Robust and rapid clustering of kpis for large-scale anomaly detection," in *26th IEEE/ACM International Symposium on Quality of Service, IWQoS 2018, Banff, AB, Canada, June 4-6, 2018*. IEEE, 2018, pp. 1–10.

[21] S. Zhang, D. Li, Z. Zhong, J. Zhu, M. Liang, J. Luo, Y. Sun, Y. Su, S. Xia, Z. Hu, Y. Zhang, D. Pei, J. Sun, and Y. Liu, "Robust system instance clustering for large-scale web services," in *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*. ACM, 2022, pp. 1785–1796.

[22] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[23] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 2014, pp. 3320–3328.

[24] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu, "Time series data augmentation for deep learning: A survey," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*. ijcai.org, 2021, pp. 4653–4660.

[25] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019, pp. 8024–8035.

[26] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016*. USENIX Association, 2016, pp. 265–283.

[27] S. Hawkins, H. He, G. Williams, and R. Baxter, "Outlier detection using replicator neural networks," in *International Conference on Data Warehousing and Knowledge Discovery*. Springer, 2002, pp. 170–180.

[28] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," in *Anomaly Detection Workshop at 33rd International Conference on Machine Learning*, 2016.

[29] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "A transformer-based framework for multivariate time series representation learning," in *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*. ACM, 2021, pp. 2114–2124.

[30] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 11 106–11 115.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "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*, 2017, pp. 5998–6008.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778.

[33] L. J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *CoRR*, vol. abs/1607.06450, 2016.

[34] Y. Dong, J. Cordonnier, and A. Loukas, "Attention is not all you need: pure attention loses rank doubly exponentially with depth," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 2793–2803.

[35] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press, 2018, pp. 3538–3545.

[36] X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, and J. Pei, "AM-GCN: adaptive multi-channel graph convolutional networks," in *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*. ACM, 2020, pp. 1243–1253.

[37] Huber and J. Peter, "Robust estimation of a location parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.

[38] (2021) 3-sigma rule. [Online]. Available: https://en.wikipedia.org/wiki/68-95-99.7_rule

[39] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Söderström, "Detecting spacecraft anomalies using lstms and non-parametric dynamic thresholding," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. ACM, 2018, pp. 387–395.

[40] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," in *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, ser. ACM International Conference Proceeding Series, vol. 148. ACM, 2006, pp. 233–240.

[41] N. Tatbul, T. J. Lee, S. Zdonik, M. Alam, and J. Gottschlich, "Precision and recall for time series," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018, pp. 1924–1934.

[42] (2021) Signal-to-noise ratio. [Online]. Available: https://en.wikipedia.org/wiki/Signal-to-noise_ratio

[43] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. ACM, 2019, pp. 2041–2055.

[44] Z. He, P. Chen, X. Li, Y. Wang, G. Yu, C. Chen, X. Li, and Z. Zheng, "A spatiotemporal deep learning approach for unsupervised anomaly detection in cloud systems," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[45] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. F. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, Ç. Gülçehre, H. F. Song, A. J. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. R. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," *CoRR*, vol. abs/1806.01261, 2018.