TimeSeriesBench: An Industrial-Grade Benchmark for Time Series Anomaly Detection Models

Haotian Si[†*], Jianhui Li[†✉], Changhua Pei[†‖], Hang Cui[‡], Jingwen Yang[†*], Yongqian Sun[§]
Shenglin Zhang[§], Jingjing Li[†], Haiming Zhang[†], Jing Han[**], Dan Pei[¶], Gaogang Xie[†]
[†] Computer Network Information Center, CAS, Beijing, China, [§] Nankai University, Tianjin, China
[*] University of Chinese Academy of Sciences, Beijing, China, [‡] Jilin University, Changchun, China
[‖] Hangzhou Institute for Advanced Study, University of Chinese Academy of Sciences, Hangzhou, China
[¶] Tsinghua University, Beijing, China, [**] ZTE Co., Ltd, China
{htsi, lijh, chpei}@cnic.cn

Abstract—Time series anomaly detection (TSAD) has gained significant attention due to its real-world applications to improve the stability of modern software systems. However, there is no effective way to verify whether they can meet the requirements for real-world deployment. Firstly, current algorithms typically train a specific model for each time series. Maintaining such many models is impractical in a large-scale system with tens of thousands of curves. The performance of using merely one unified model to detect anomalies remains unknown. Secondly, most TSAD models are trained on the historical part of a time series and are tested on its future segment. In distributed systems, however, there are frequent system deployments and upgrades, with new, previously unseen time series emerging daily. The performance of testing newly incoming unseen time series on current TSAD algorithms remains unknown. Lastly, the assumptions of the evaluation metrics in existing benchmarks are far from practical demands. To solve the above-mentioned problems, we propose an industrial-grade benchmark TimeSeriesBench. We assess the performance of existing algorithms across more than 168 evaluation settings and provide comprehensive analysis for the future design of anomaly detection algorithms. An industrial dataset is also released along with TimeSeriesBench.

Index Terms—Anomaly Detection, Univariate Time Series, Deep Learning, Benchmark

I. INTRODUCTION

In recent years, we have witnessed the rapid growth in the scale of software systems and services. To ensure service quality assurance and maintain user satisfaction, IT operation engineers must monitor the time series metrics like response time or success rate to judge if there are system failures. Time series anomaly detection (TSAD) aims to identify irregular patterns in these time series data, which indicate potential or existing system failures. The outliers can often signify critical deviations from the norm, such as impending machine failures in industrial processes [1] and malignant network traffic attacks in web applications [2]. Accurate and timely anomaly detection can help to reduce the mean time to repair, reduce the loss in revenue and maintain the reputation and branding for a company.

Owing to the clear significance and applicability of TSAD in these real-world scenarios, in recent years, a variety of anomaly detection methods, particularly those based on deep learning, are burgeoning incessantly [3]–[10]. Nonetheless, there is a considerable divergence across the results reported by different papers even on the same dataset, as they employed various evaluation criteria and learning schemas. Despite the existence of some benchmarks for anomaly detection algorithms [11]–[16], they struggle to provide practical guidance for industrial domain experts to apply and develop deep learning methods as they pay more attention to statistical methods. In general, applying TSAD algorithms to practical systems faces these main obstacles: **(I)** Training specific model for each time series and deploying an exclusive model for each time series results in unaffordable maintenance costs. **(II)** New systems are more likely to encounter issues when first launched or upgraded, which makes anomaly detection even more essential despite their almost non-existent volume of historical data. **(III)** Existing metrics hold one-sided assumptions for evaluating how well the algorithms perform, which cannot offer an effective reference for industrial practice. **(IV)** There is no platform continuously integrating new methods and comparing them in a unified and intuitive manner, similar to the GLUE Leaderboard in NLP [17], which prevents experts in the industry from keeping pace with the latest algorithms and advancements in academia.

In response to the four challenges mentioned earlier, we propose **TimeSeriesBench**, an industrial-grade benchmark for evaluating time series anomaly detection. This benchmark has four main features: **(I)** To address the issue of high maintenance costs and non-deployability caused by existing works requiring a separate model for each time series, we adopt an All-in-One training paradigm to assess the detection performance of current algorithms when only one unified model is trained. **(II)** To tackle the inability of existing works to handle new time series, we employ a Zero-Shot inference paradigm during evaluation. By using a novel data-splitting method, we assess the model's detection performance on previously unseen curves without retraining or fine-tuning new time series. **(III)** We thoroughly integrate existing evaluation criteria, and propose our own event-based evaluation metric to ensure that benchmark results can provide effective guidance
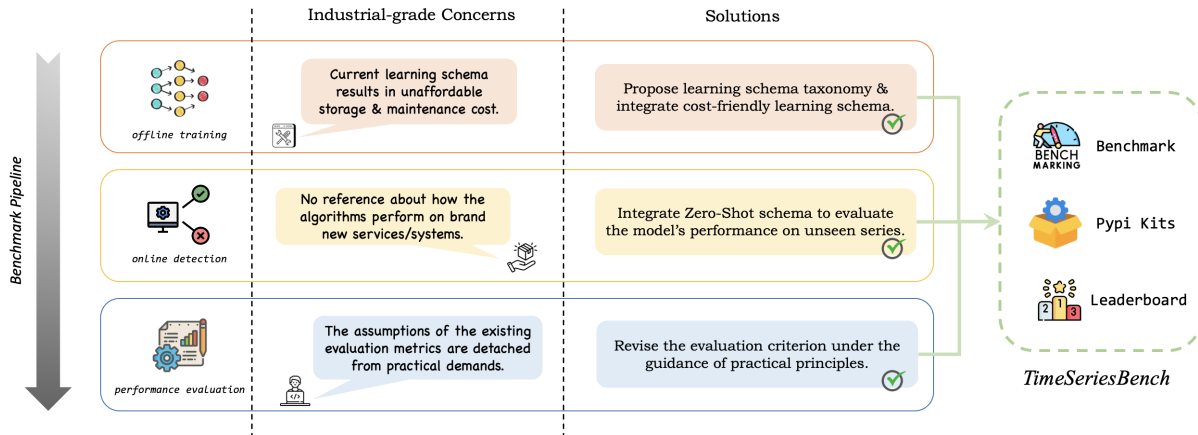
Fig. 1: The industrial-grade concerns and TimeSeriesBench's solutions on benchmarking univariate time series anomaly detection algorithms.

for industrial deployment. Also, TimeSeriesBench provides a series of well-known metrics, which can meet the detection needs of different downstream application scenarios. **(IV)** To solve the problem of industry experts not having access to a unified and continuously updated set of evaluation results, we have built an online leaderboard. This leaderboard combines the above three features and conducts a comprehensive evaluation across more than 168 evaluation settings in terms of training and testing paradigms, evaluation metrics, and multiple datasets.

Based on TimeSeriesBench, we evaluate several representative time series anomaly detection methods ranging from statistical machine learning and deep learning to large time series models. From this, we have made some novel observations, conducted detailed analyses, and offered insights for algorithm design and deployment. To name a few, models that employ variational autoencoder exhibit good detection performance for pattern-wise anomalies, and so far the general time series models struggle to outperform methods specifically designed for anomaly detection tasks.

The paper's main contributions are as follows:

- We present the first online leaderboard for time series anomaly detection algorithms, which upgrades the existing evaluation frameworks across multiple dimensions such as training, inference, evaluation and datasets. It effectively supports the industry experts' need to select the best academic algorithms and provides an industrial-grade evaluation method for academic algorithms, ensuring the deployability of future algorithms.
- For the first time, we employ the cost-friendly all-in-one and zero-shot settings to evaluate several well-known state-of-the-art (SOTA) anomaly detection methods on TimeSeriesBench and discover some enlightening conclusions, providing directions for future optimization.
- We develop and publish a comprehensive evaluation toolkit built with Python named EasyTSAD, providing a one-stop solution for data processing, model training,

and assessment, which we have made open source to the community to accelerate the efficiency of existing anomaly detection algorithm optimizations.
- To address the issue of inaccurate anomaly labeling in existing public datasets [18], we collaborated with a global large company and invited business system experts to meticulously annotate anomalies in the online system. After detailed calibration, we have also made the dataset publicly available to the community as part of Time-SeriesBench, supplementing existing anomaly detection datasets. The codes, data and the online leaderboard are released publicly[1].

II. PRELIMINARIES

A. Time Series Anomaly Detection

A time series consists of successive observations of a metric (e.g., queries per second, sensor value, etc.) over a long period of time. The series can be represented as $T = x_1, x_2, \cdots, x_n$, where x_i represents an observation at timestamp i, and n denotes the length of series. An anomaly within these time series data can be identified as a single point or a sequence of points that diverge significantly from their former customary patterns observed in the sequence. Anomaly detection methods project sequence observations into a probability distribution space to represent the degree of anomaly of the current observation, namely *anomaly score*, and compare it with a predefined threshold to determine whether an anomaly has occurred. Due to the scarcity of anomaly labels, existing deep-learning methods often opt for self-supervised training approaches.

B. Anomaly Types

Following the behavior-driven taxonomy [13], anomaly types can be roughly divided into point-wise outliers and

---

[1]The code is available at https://github.com/CSTCloudOps/EasyTSAD. The new proposed dataset is available at https://github.com/CSTCloudOps/Dataset-for-TSAD, and the online leaderboard is available at https://adeval.cstcloud.cn.

pattern-wise outliers (Fig. 2). Point-wise outliers denote unexpected incidents like spikes or glitches on individual time points or within very short periods of time. Pattern-wise outliers represent anomalous subsequences that span over a certain period of time, often characterized as discordances or inconsistencies within the data.
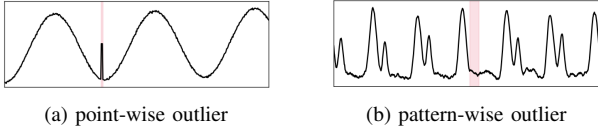

(a) point-wise outlier   (b) pattern-wise outlier

Fig. 2: Illustrations of anomaly types. Anomalous segments are highlighted in pink.

III. TimeSeriesBench Settings

We launch TimeSeriesBench to integrate various aspects that are widely concerned into a pipeline to meet the diverse needs of scientific research and practical applications, and further promote the development of communities related to the field of time series anomaly detection. Aside from model implementation, the platform also takes some controversial issues, for instance, learning schema and evaluation criteria into consideration, to provide the researchers and engineers with a more comprehensive view of whether the models perform well in specific scenarios. In this section, we will illustrate the motivation and the implementation details of each inductive setting.

A. Datasets

High-quality datasets are the prerequisite for effective training and reasonable performance evaluations for models. Unfortunately, several publicly available time series datasets are claimed flawed due to their unrealistic anomaly density, or mislabeled ground truth [18]. Moreover, the proportion of different types of outliers can vary widely among datasets as the time series are gathered from irrelevant domains and applications. For example, Yahoo [19] is dominated by global and contextual point-wise outliers on the basis of the behavior-driven taxonomy proposed by [13] (the trivial first-order difference can achieve a comparable performance), while pattern-wise outlier plays a significant role in UCR archive [20] (shown in Sec. V). To this end, we gather multiple well-labeled real-world datasets covering many application domains to diversify the distributions of anomalies, meanwhile excluding markedly flawed datasets (some cases in [18]). Furthermore, we introduce a synthetic dataset generated from TODS [13] for the convenience of specific case analysis due to its good interpretability, because the time series adheres to well-designed distributions.

Real-world datasets. We employ AIOPS [21], WSD [22], Yahoo [19], NAB [23], and UCR [20] for evaluations. Real-world datasets are more susceptible to uncertainties and tend to mix up with even inexplicable noise, which raises the demand in terms of model robustness. As the classes of anomalies are imbalanced among these datasets, practitioners focusing on the

general-purpose model should refer to detection performance on *all* datasets, while application-oriented tasks need to pay more attention to performance on datasets that are strikingly similar to specific scenes. Meanwhile, we release a new dataset collected from the production environment, called NEK (Network Equipment KPI), for a more comprehensive evaluation.
Synthetic dataset. TODS [13] introduces a novel taxonomy, categorizes anomalies into five types, and publishes an anomaly generation toolkit in line with their claims. We conduct modifications based on their codes to generate longer and non-trivial anomalies, which would allow researchers to determine if models are capable of particular types of anomalies in a straightforward manner.

B. Learning Schema

Existing benchmarks take it for granted that the training process of the detector should comply with the naive task-specific schema, more specifically, training an exclusive detector for each time series solely leveraging its own historical patterns. We claim that it's time to break the prevailing stereotype, especially for deep learning models, given the great transformations observed in application scenarios during recent years. On the one hand, with the maturation and improvement of surveillance systems in various fields, the quantity of time series to be monitored has experienced a substantial increase, reaching several orders of magnitude higher than before. This significantly raised the storage overheads if detectors are exclusive. On the other hand, the previous years witnessed many seminal works pertinent to foundational models for time series forecasting or other general tempo tasks [24]–[27]. We aspire to establish a foundation for the standardized training and evaluation of large-scale, general-purpose time series models, especially those designed for anomaly detection. Thus we introduce two novel learning schemas, called all-in-one mode and zero-shot mode, to assess the performance variations of models in scenarios involving large-scale data or zero-shot settings.
Naive schema. In this mode, we input a single time series for model training/fitting, and the trained detector is specifically employed for online detection on that particular series. Intuitively, this facilitates the model to make a more precise description of the temporal pattern given enough data. Nevertheless, it is worth noting that the volume of a single series tends to be insufficient.
All-in-one schema. In this mode, only one unified model instance is trained using all the sequences in the dataset, and the trained model is then applied for real-time anomaly detection across all sequences within the dataset. This schema exposes more patterns embedded in various series to the model, thereby providing an additional opportunity for the model to learn common and inherent traits shared among the time series. However, due to the differing definitions of anomalies across series, this carries the risk of confounding the model with conflicting information when detecting anomalies online (i.e., an anomaly present in a specific curve may not necessarily be considered an anomaly in other curves). Several
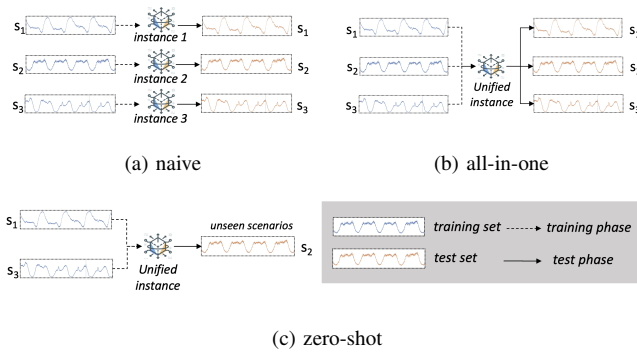
(a) naive

(b) all-in-one

(c) zero-shot

Fig. 3: Illustrations of three learning schemas. s_n indicates the n-th time series in the dataset.

novel methods [28], [29] have adopted this schema driven by practical needs.

Zero-shot schema. In zero-shot mode, the whole dataset is split into two disjoint subsets. One subset is employed for model training and the other is used to evaluate the detection performance. This schema has been devised in light of practical considerations. Specifically, it addresses situations where a system is newly deployed with no prior historical data and a robust and adaptable model is required to navigate through this gap. This places higher demands on the model's ability to capture the intrinsic representations of time series.

C. Evaluation Criteria

A flawless evaluation criteria serves as the foundation for not only assessing the effectiveness of methods but also guiding model parameters optimization. Nevertheless, recent research has uncovered substantial limitations in commonly employed evaluation criteria, on which some seemingly absurd methods (e.g. noise generated by Random Guess) can outperform all others [30]–[33]. Also, some assumptions employed by certain evaluation criteria also conflict with the principles of industrial practice. Therefore, it is imperative to revise an anti-cheating and industrial-oriented evaluation criterion for our benchmark. *It is crucial to emphasize that the selection of criterion is closely tied to the specific application context and determines how models perform with regard to the aspect you are interested in.* In this paper, we focus on a generic *real-time* anomaly detection task.

In industrial practice, we prefer a criterion that aligns better with the typical workflow of real-time anomaly detection systems deployed in real-world scenarios. In a practical software system, whenever the anomaly score exceeds a predefined threshold, the system triggers an alert to notify the operators. Therefore, an operator tends to prioritize the following key issues:

a. If a method can always detect anomalies within the anomaly segments, even if the anomaly score surpasses the threshold only once in each segment, it is still considered to possess a significant recall capability.

b. Excessive false alarms would be rather frustrating for operators as they are obligated to examine each alert.

c. The anomaly that lasts longer is likely to be more severe or challenging than the shorter ones.

d. Detecting and addressing anomalies as early as possible can significantly mitigate the economic losses caused by abnormal situations.

Unfortunately, current evaluation criteria [14], [30], [32], [34]–[36] fail to simultaneously address all of the above issues. Thus we propose a new criterion that eliminate the effect of the factors that distort the evaluation results by the following strategies:

Point Adjustment. We utilize a strategy of "adjusting" the output of the algorithm, namely point-adjustment (PA), as the solution to issue *a*. This strategy is widely adopted [1], [9], [28], [37]–[40]. Under this strategy, all timestamps within an anomalous segment are assigned the highest anomaly score present within that segment, thus the whole anomaly segment is considered to be detected if at least one anomaly score surpasses the threshold. Then the F1 score is obtained in a point-based manner (point-wise PA in Fig. 4).

Anomaly Weights Revision. However, the naive point-wise PA introduces biased true positives and false negatives. As shown in Fig. 4, as a long anomaly segment is detected under point-adjustment, even if two false alarms are triggered, the Precision is up to 0.8, which contradicts the demand corresponding to issue *b*. The intrinsic flaw of this criteria is that the detection is rewarded generously while pulsing false alarm is penalized just once [30]. Garg et al. [33] revise the primitive PA from an event perspective. Each anomaly segment is treated as an individual event and contributes to a true positive or false negative only once. This criterion, namely event-wise PA, prevents the inflated evaluation scores originated from illogical criteria, while absolutely overlooking the length of the anomaly segments. Based on the observation *c* , we apply a severity coefficient $\ln(k + e)$ to adjust the true positive and false positive measurements associated with the given anomalous segment of length k. The criterion is called reduced-length PA, and we categorize it along with event-wise PA as event-based evaluation criteria.

Latency Constraint. Issue *d* is addressed by imposing strict constraints on the detection latency. As depicted in the illustration (Fig. 5), assuming the latency limit (k) is set to 3, an anomaly is considered effectively detected only if it is identified within three sampling points after its occurrence. We designate this strategy as *k-delay adjustment*. This measure enables a more precise assessment of whether the model can meet the requirement of the scenario where there is a high demand for real-time responsiveness. It is equally essential to acknowledge that this approach is applicable only to datasets whose anomalies are labeled without positional bias. Unfortunately, not all datasets can meet this requirement, thus we present the related results as supplementary content.

Anomaly Lag Elimination. Most statistical and deep learning methods generate anomaly scores under prediction-based or reconstruction-based frameworks, both heavily relying on
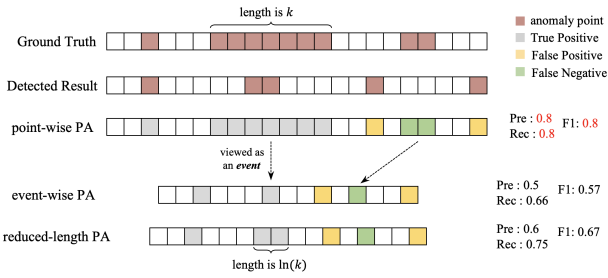
Fig. 4: Illustrations of evaluation criteria based on point-adjustment (PA). Point-wise PA gives an inflated score when some anomaly segments persist for a long duration. Event-wise PA treats each anomaly segment as an event, completely disregarding the length of the anomaly segment. Reduced-length PA considers the trade-offs between the two methods, holding greater practical significance in real-world applications.
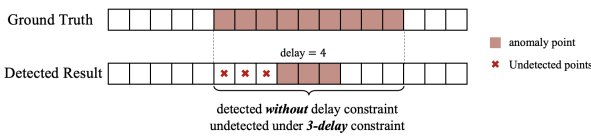

Fig. 5: Illustrations of k-delay adjustment. This strategy can be combined with point-adjustment as a complementary evaluation paradigm.

the pattern provided by previous time windows. Hence, the recently concluded anomaly segment, particularly those with longer durations, has the potential to heavily interfere with the subsequent detection process. As shown in Fig. 6, from a qualitative perspective, we observe that the method accurately detects the frequency anomaly *event*. However, some unexpected false positives emerge due to the aforementioned reasons, resulting in an underestimated evaluation. We slightly extended the anomaly segments (less than 10 time points) to tolerate such occurrences, to make the evaluation more in line with our intuitive comprehension. We carefully handle edge cases to avoid merging two anomaly segments during the operation.
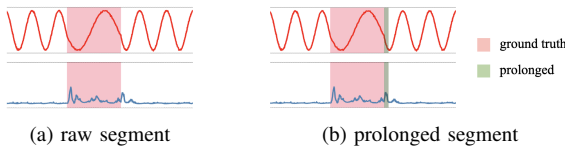

Fig. 6: Illustrations of prolonging the anomaly segments. The blue line denotes the anomaly scores provided by a particular method. Considering the interference caused by anomalies on the detection of normal values, we should exclude false positives that occur immediately after an anomaly segment.

D. Algorithms

We adopt 17 semi-supervised/unsupervised methods that have raised significant discussion and wide-ranging impact in

the field of univariate time series anomaly detection. From the perspective of method attributes, these methods can be divided into statistical methods and deep learning methods.

Statistical methods. These methods, including Sub-LOF [41], SAND [42], and MatrixProfile [43], directly compute the minimum similarity between the current and the historical time window. This manner, however, would make the model highly sensitive to the data characteristics in specific scenarios, resulting in a lack of generalizability.

Deep learning methods. These methods learn the traits of normal patterns from past data in the training phase and judge whether the current metric is within the normal range in the online detection phase. Furthermore, deep learning methods can be categorized into the following families:

Prediction-based. These methods aim to predict the "normal" value of the current metric according to the adjacent observations. If the ground truth is far from the predicted value, an anomaly alarm will be triggered. The representative works include AR [44], LSTMAD[2] [45],

Reconstruction-based. These methods aim to denoise the anomaly points by the encoding-decoding phase. If the reconstructed value is far from the ground truth, it can be assumed that an anomaly occurs. The representative works include AE [46], EncDecAD [47], SRCNN [29], Anomaly Transformer [3], TFAD [28], and TranAD [48].

VAE-based. These methods hold the assumption that the distribution mapping and sampling process can enhance the denoising effect of the model and learn a more robust representation of normal patterns. The representative works include Donut [4] and FCVAE [49].

General Time Series Model. The designed universal model structure in this class can be used for all time series tasks including time series forecasting, classification, and anomaly detection. They tend to introduce a more general inductive bias that aims to better represent the general features of time series. The representative works include TimesNet [50], OFA [51] and FITS [52].

IV. EXPERIMENTAL SETTINGS

A. Experimental Platform

All experiments are performed on a server equipped with Dual Intel(R) Xeon(R) Silver 4316 (12-core) and 256GB RAM. The operating system of this server is Ubuntu 22.04 LTS. An NVIDIA GeForce RTX 3090 24GB GDDR6 GPU is utilized to accelerate the training and inference processes of all models.

B. Datasets

With the exception of the UCR dataset and AIOPS dataset where the training and testing sets are already specified, we partition each time series into training, validation, and test sets following a 4:1:5 ratio. Given that anomalies in some datasets are randomly distributed, the test set of some sequences does

---

[2]As there is no source code available, we implement LSTMAD in two forms: LSTMAD-α in a seq2seq manner and LSTMAD-β in a multi-step prediction manner.

not contain any anomalies after partitioning. Therefore, we exclude these anomaly-free sequences from consideration. Due to the fact that the original implementation of TODS sometimes generates anomalies that do not match their specified anomaly types (e.g., constructing trend and seasonal anomalies may result in obvious global outliers), we modify the anomaly generation code in TODS to ensure that the injected anomalies align more closely with their defined types. It is worth noting that we aggregate the overall evaluation at the dataset level instead of the curve level because the imbalance in sample sizes across different datasets can lead to biased results.

C. Learning Schema

We evaluated existing methods under all proposed learning schemas to obtain a more comprehensive understanding of the model's performance from different perspectives. Statistical methods are not assessed under all-in-one and zero-shot schema due to assumption conflict. Under the all-in-one schema, taking the UCR dataset as an example, we mix all samples together from all time series' training sets during the training phase. Under zero-shot schema, we use a fixed random seed to split UCR into two subsets, each of which includes half of the time series. We mix all training samples from one subset, and the other acts as the test set. All methods share the same training set in zero-shot mode.

D. Evaluation Metrics

Following the event-based evaluation criterion outlined in Sec. III-C, we establish two concrete metrics, namely $F1_{best}$ and $AUPRC$. $F1_{best}$ denotes the highest F1 score calculated under reduced-length point-adjustment when iterating over all possible thresholds. $AUPRC$ calculates the area under the precision-recall curve generated according to reduced-length point-adjustment, which is widely applied in cases where class imbalance is present in the data. While $F1_{best}$ can measure the best detection performance that the model can achieve on the current test set, $AUPRC$ provides a more nuanced evaluation of the model's performance across different levels of recall, which can be important in anomaly detection. A model with a higher $AUPRC$ tends to be more robust.

E. Implementation details of Algorithms

All methods are implemented in Sklearn or Pytorch, either based on open-source repositories[3] or reproduced based on the original paper's description. All methods are integrated into the TimeSeriesBench suite. If the model has specific hyperparameters set for a particular dataset, we use the parameters specified for that dataset. Otherwise, we use the default hyperparameters

[3]SRCNN comes from https://github.com/microsoft/anomalydetector, AnomalyTransformer comes from https://github.com/thuml/Anomaly-Transformer, TimesNet comes from https://github.com/thuml/Time-Series-Library/blob/main/models/TimesNet.py, Donut comes from https://github.com/wagner-d/TimeSeAD/blob/master/timesead/models/generative/donut.py, TFAD comes from https://github.com/DAMO-DI-ML/CIKM22-TFAD, OFA comes from https://github.com/DAMO-DI-ML/NeurIPS2023-One-Fits-All, FITS comes from https://github.com/VEWOXIC/FITS,

provided in the source code. The early stopping mechanism is applied to all methods on the validation set.

V. Experimental Results

In this section we conduct a rigorous analysis of the model's effectiveness under various settings, aiming to provide meaningful research insights (**RI**) from unprecedented perspectives that can have implications for the design and application of methods. We aim to provide insights into the following research questions:

1. How is the overall performance of the models, and what factors contribute to these results?
2. How does the model's performance vary under different learning schemas?
3. How does the model's performance vary in detecting different types of anomalies?

A. Overall Performance

The overall ranking list is presented in Fig. 7, including the performance of each method using naive, all-in-one, and zero-shot schema, respectively. Since statistical methods, especially those based on streaming, heavily rely on the premise of data being identically distributed, these methods are not well-suited for the all-in-one and zero-shot learning schemas. Therefore, we only evaluate them under the naive schema. First and foremost, the performance of various models under the naive and all-in-one learning schema can exhibit substantial disparities (**RI 1**). For example, on the NEK dataset, although Donut performs well on the naive schema, its performance significantly degrades under the all-in-one schema. In contrast, FITS achieve better performance under all-in-one schema on this dataset. We list the impacts of different learning schemas on the performance of different methods in Table I. *It is thought-provoking that these differences do not show a clear bias, suggesting that these variations are the result of a combination of multifaceted factors, which is discussed in detail in Sec. V-B.* Additionally, given the consideration of errors arising from random dataset partitioning, the models demonstrated a relatively consistent performance across the all-in-one and zero-shot modes, as there may exist interdependence among time series in the same dataset (**RI 2**).

Generally, the statistical methods exhibit great performance on the dataset with relatively stable shapelets and low noise level (UCR), but work poor on other datasets with more noise. For deep learning based models, contrary to our expectations, the overall performances of the models that contain more complicated structures are not satisfactory (**RI 3**). We present a case (Fig. 8) from the AIOPS dataset to vividly demonstrate this phenomenon. Even though these point-wise anomalies seem to be quite evident, a significant portion of the methods are unable to handle such events effectively. As numerous factors can lead to a decrease in performance for DNNs, we summarize the major factors that result in the underperformance of some deep learning methods in anomaly detection tasks:

(a) naive	AIOPS	NAB	TODS	WSD	Yahoo	UCR	NEK
AR	0.75	0.90	0.75	0.87	0.92	0.45	0.96
LSTMAD-α	0.81	0.90	0.70	0.94	0.55	0.48	1.00
LSTMAD-β	0.80	0.90	0.76	0.94	0.54	0.44	0.97
FCVAE	0.79	0.87	0.71	0.91	0.69	0.59	0.90
AE	0.72	0.88	0.68	0.92	0.61	0.42	0.98
FITS	0.71	0.86	0.54	0.88	0.76	0.41	0.90
OFA	0.69	0.89	0.63	0.88	0.75	0.42	0.95
EncDecAD	0.76	0.88	0.64	0.93	0.50	0.32	0.76
Donut	0.70	0.84	0.72	0.88	0.68	0.44	0.99
TranAD	0.65	0.91	0.25	0.67	0.56	0.25	0.91
TimesNet	0.67	0.84	0.37	0.87	0.44	0.25	0.93
TFAD	0.36	0.57	0.45	0.76	0.78	0.34	0.62
SRCNN	0.16	0.58	0.30	0.18	0.13	0.32	0.44
Anomaly Transformer	0.37	0.77	0.28	0.30	0.14	0.31	0.40
MatrixProfile	0.03	0.18	0.21	0.04	0.21	0.51	0.32
SAND	0.07	0.30	0.25	0.10	0.47	0.55	0.18
Sub-LOF	0.38	0.70	0.58	0.72	0.49	0.58	0.68

(b) all-in-one	AIOPS	NAB	TODS	WSD	Yahoo	UCR	NEK
AR	0.82	0.89	0.79	0.93	0.95	0.46	0.96
LSTMAD-α	0.81	0.85	0.75	0.95	0.80	0.61	0.93
LSTMAD-β	0.82	0.85	0.75	0.94	0.81	0.61	0.96
FCVAE	0.76	0.89	0.85	0.81	0.85	0.48	0.81
AE	0.74	0.84	0.77	0.92	0.82	0.36	0.96
FITS	0.70	0.87	0.71	0.88	0.93	0.43	0.96
OFA	0.55	0.86	0.62	0.81	0.77	0.29	0.76
EncDecAD	0.70	0.83	0.65	0.84	0.58	0.33	0.95
Donut	0.58	0.80	0.84	0.77	0.65	0.40	0.58
TranAD	0.66	0.86	0.56	0.74	0.52	0.25	0.96
TimesNet	0.50	0.85	0.50	0.54	0.53	0.19	0.93
TFAD	0.49	0.47	0.56	0.80	0.94	0.49	0.41
SRCNN	0.67	0.64	0.26	0.77	0.17	0.38	0.62
Anomaly Transformer	0.37	0.73	0.26	0.58	0.24	0.24	0.67

(c) zero-shot	AIOPS	NAB	TODS	WSD	Yahoo	UCR	NEK
AR	0.80	0.90	0.80	0.93	0.94	0.49	0.96
LSTMAD-α	0.82	0.86	0.68	0.93	0.83	0.59	0.95
LSTMAD-β	0.81	0.86	0.59	0.94	0.79	0.56	0.96
FCVAE	0.80	0.94	0.80	0.77	0.80	0.41	0.80
AE	0.77	0.85	0.71	0.90	0.86	0.32	0.96
FITS	0.67	0.88	0.66	0.88	0.93	0.42	0.96
OFA	0.55	0.85	0.65	0.85	0.83	0.35	0.82
EncDecAD	0.75	0.85	0.71	0.83	0.60	0.32	0.94
Donut	0.69	0.72	0.62	0.79	0.70	0.41	0.62
TranAD	0.47	0.93	0.40	0.62	0.49	0.29	0.89
TimesNet	0.49	0.88	0.57	0.62	0.45	0.26	0.72
TFAD	0.47	0.28	0.64	0.89	0.93	0.47	0.45
SRCNN	0.69	0.64	0.30	0.74	0.14	0.36	0.53
Anomaly Transformer	0.44	0.56	0.22	0.30	0.43	0.27	0.54

Fig. 7: Overall performance ranking using different learning schema under reduced-length $F1_{best}$. Methods are ranked in descending order according to the average of all scores. The row names denote the names of methods, while the column names denote the names of the datasets. The best score of each column is underlined. Due to the inherent assumptions behind the statistical methods being incompatible with the all-in-one and zero-shot schemas, these methods are not included in the ranking, and the related score cells are set to grey without values.
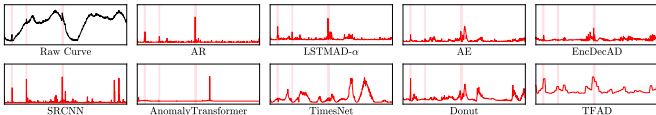

Fig. 8: Illustrations that some methods fail to effectively detect point-wise anomalies in certain scenarios. The black line represents the original curve, while the red line represents the anomaly scores provided by the method. Anomalous segments are highlighted in pink.

Poor noise resistance. The models' anti-noise ability plays an important role in AD tasks. Compared to other domains like NLP or CV, time series data is affected by more factors during the collection process, resulting in more and intractable noise. Although models using simple structures (AR, FITS) sacrifices some feature expression capability, it's hard for them to overfit noise. This contributes to the good results. For LSTM-series methods, due to the iterative encoding process, the forget gate in LSTM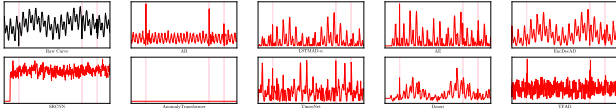s is more likely to minimize the retention of low signal-to-noise ratio data in the hidden state. In contrast, due to the lack of special design for anti-noise, simply employing complicated backbones like transformer may lead to the overfitting of the noise, and finally result in the poor performance as shown in Fig. 8 **(RI 4)**.

Lack of training data. As shown in Fig. 9, some models, especially the ones with complicated structures, achieve significant improvement in performance when using more training data. Models with complicated structures usually require a larger volume of training data to avoid overfitting due to their higher degrees of freedom and flexibility **(RI 5)**. Similarly, models that employ intricate loss functions often necessitate a greater amount of training data, including diverse examples, to prevent the emergence of falling into local minima and even trivial solutions.
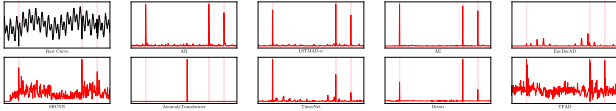
Trends with large periods. Some data may exhibit long-term trends or patterns that span large periods. These trend components can cause the data distribution to deviate from the comfort zone of the model, causing the model to calculate anomaly scores based on a biased data distribution it has never

TABLE I: Performance differences on all datasets using different learning schemas. Best scores are highlighted in bold, and second-best scores are highlighted in bold and underlined. Performance improvements exceeding 5% are marked with ↑, while performance declines exceeding 5% are marked with ↓.

Method	AIOPS		NAB		TODS		WSD		Yahoo		UCR		NEK	
	S_{one}	S_{all}	S_{one}	S_{all}	S_{one}	S_{all}	S_{one}	S_{all}	S_{one}	S_{all}	S_{one}	S_{all}	S_{one}	S_{all}
AR	0.7517	**0.8214**↑	**_0.9035_**	**0.8881**	**_0.7472_**	0.7859↑	0.868	0.9276↑	**0.9195**	**0.948**	0.4454	0.4636	0.9555	**0.9612**
LSTMAD-α	**0.8081**	0.8141	0.8984	0.8492↓	0.7015	0.7538↓	**0.9381**	**0.9467**	0.5496	0.8↑	**_0.4836_**	**0.6144**↑	**0.9971**	0.9349↓
LSTMAD-β	**_0.8034_**	_0.8151_	0.9031	0.8484↓	**0.7589**	0.7495	**0.939**	**_0.9445_**	0.5366	0.8114↑	0.4366	**_0.6101_**↓	0.9744	0.961
AE	0.7222	0.7448	0.8751	0.8375	0.6842	0.7712↑	0.918	0.9192	0.612	0.8205↑	0.4198	0.3644↓	0.9801	0.9609
EncDecAD	0.7583	0.7003↓	0.8752	0.8336	0.6359	0.6534	0.9285	0.8353↓	0.5018	0.5842↑	0.3178	0.3333	0.755	0.9534↑
Donut	0.6957	0.5827↓	0.8397	0.7966↓	0.7207	**_0.836_**↑	0.8787	0.7666↓	0.6813	0.6498	0.4409	0.4↓	**_0.9885_**	0.5801↓
FCVAE	0.7851	0.7593	0.874	**_0.8857_**↓	0.7145	**0.8526**↑	0.9087	0.8121↓	0.6883	0.8537↑	**0.5873**	0.4766↓	0.8977	0.8148↓
SRCNN	0.1627	0.6672↑	0.5802	0.635↑	0.3042	0.2598↓	0.1785	0.7742↑	0.1261	0.1728↓	0.3227	0.3791↓	0.4365	0.6173↑
AnomalyTransform	0.3728	0.3656	0.7739	0.7322↓	0.2827	0.2631↓	0.2984	0.5838↑	0.145	0.2377↑	0.3104	0.2372↓	0.3997	0.666↑
TFAD	0.3551	0.4889↑	0.5746	0.4679↓	0.4526	0.5626↑	0.7561	0.8016↑	**_0.781_**	**_0.9361_**↑	0.3398	0.4941↑	0.625	0.4103↓
TranAD	0.6486	0.6561	**0.9101**	0.8567↓	0.2531	0.5624↑	0.6738	0.7409↑	0.562	0.5211↓	0.2527	0.2483	0.9071	0.9571↑
TimesNet	0.6737	0.4988↓	0.8419	0.8544	0.3695	0.5032↑	0.8684	0.539↓	0.4444	0.5341↑	0.2493	0.1856↓	0.9335	0.9257
OFA	0.6891	0.5544↓	0.8869	0.861	0.6263	0.6187	0.8784	0.8111↓	0.7512	0.766	0.4169	0.2888↓	0.9471	0.7642↓
FITS	0.7139	0.6986	0.8617	0.8683	0.5371	0.7146↑	0.8796	0.8788	0.7564	0.9261↑	0.4092	0.428	0.8975	**0.9625**↑

(a) Lacking data under one-by-one schema

(b) Enriching data by using all-in-one schema

Fig. 9: Case study on Yahoo_A3Benchmark-TS13 shows that lack of training data induces poor performance.
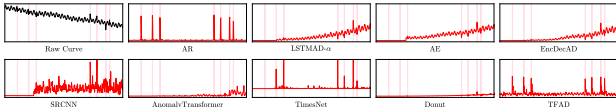
encountered before **(RI 6)**.

Fig. 10: Case study on Yahoo_A3Benchmark-TS14 shows that trends with large periods induce poor performance. The scores are obtained under the all-in-one learning schema, eliminating the impact of insufficient data.

Unreasonable inductive bias. Deep learning models rely on their architectural design and optimization algorithms to detect anomalies in a semi-supervised manner. If the chosen model architecture or optimization approach is not well-suited for point-wise anomaly detection, the models may have an unreasonable inductive bias that hampers their ability to accurately detect anomalies **(RI 7)**.

Based on the above observations, it is imperative for newly proposed methods to place greater emphasis on these factors to avoid poor results on these distinct anomalies. We believe that the potential of the powerful representation capability of complex structures has not been fully explored yet in time series anomaly detection tasks. We strongly advocate prioritizing the refinement of module design or loss function

in order to enhance the models' resilience to noise and their capacity for generalization. Thus complicated backbones can better leverage their expressive power for temporal features without overfitting to noise.

B. Performance in Fine-Grained Scenarios

Merely referring to aggregated data at the granularity of datasets is insufficient to gain a profound understanding of the model's performance. We carry out a fine-grained comparison of model performance from the perspectives of both learning schema and anomaly type. The UCR dataset is widely acknowledged for its comprehensive annotation of various anomaly types, making it a well-labeled dataset. Additionally, each time series in the UCR dataset is guaranteed to contain only one anomaly, which facilitates the categorization of different anomaly types. To further refine our analysis, we partition the UCR dataset into two subsets based on its supplemental materials [53]: one subset exclusively consisting of point-wise anomalies and another subset focusing solely on pattern-wise anomalies. We form a 2x2 matrix to reflect the variations in model performance under different conditions by incorporating the dimensions of "naive" and "all-in-one" learning schemas, along with different types of anomalies, as shown in Fig. 11.

From the boxplots, no solution can always outperform the others in all situations **(RI 8)**. It is also apparent that the majority of models exhibit a significantly higher detection performance for point-wise anomalies compared to pattern-wise anomalies regardless of the learning schema employed **(RI 9)**. On a more detailed level, prediction-based models excel in identifying point-wise anomalies, as the steep peak/valley is unpredictable in most cases **(RI 10)**. Due to the inclusion of a more diverse data distribution under all-in-one learning schema, the models are more likely to learn robust representations of patterns. As a result, under the all-in-one learning schema, there is a significant improvement in the detection performance of point-wise anomalies **(RI 11)**. This observation is particularly pronounced in the case of the Yahoo dataset each time series of which is relatively short. With regard to pattern-wise anomalies which are considered to be more challenging,

(a) naive & point-wise

(b) naive & pattern-wise

(c) all-in-one & point-wise
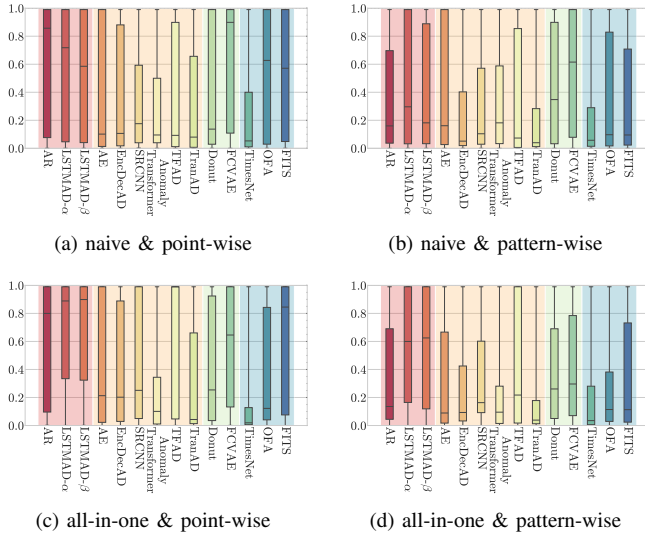

(d) all-in-one & pattern-wise

Fig. 11: The 2x2 matrix reflecting the variations in model performance under different conditions. We denote prediction, reconstruction, VAE-based, and general time series methods in light red, yellow, green, and blue backgrounds.

the situation becomes more complex and interesting. Methods that aim to generate the temporal window after projecting them to low-dimensional representations (Donut and FCVAE) surpass all others in detecting pattern-wise anomalies, while the performance greatly declines when under the all-in-one mode (**RI 12**). The former indicates that the assumption of low-dimensional representations hardly reconstructing high-level pattern-wise anomaly indeed works. An illustration is presented to confirm this in Fig. 12(a), where Donut can easily handle the "smooth" high-level anomaly. However, the performance deteriorates or even becomes ineffective under the all-in-one mode (Fig. 12(b)). We hypothesize that the capacity of low-rank representations may struggle to cover the diverse data distributions present in various curves. Also, although the general time series methods (TimesNet, OFA, FITS) perform well in other time series tasks, they struggle to outperform methods specifically designed for anomaly detection tasks like LSTMAD and FCVAE (**RI 13**). This may indicate that there is a gap between time series anomaly detection tasks and general time representation tasks. For example, anomaly detection tasks may require models to have a stronger denoising effect. These gaps need special attention when designing models for anomaly detection tasks.

C. Performance under Delay-Constraint

Issue d in Sec. III-C demands exceptionally high data set quality, whereas some datasets (including NAB, UCR, Yahoo) fail to offer labels without positional bias. Therefore, the result merely provide a approximate reference on these datasets. We manually set a relatively suitable latency limit (K) for each dataset based on the sampling frequency and label quality of the dataset to evaluate the performance of each method under strict real-time requirements. The over-

(a) Model comparison under naive learning schema

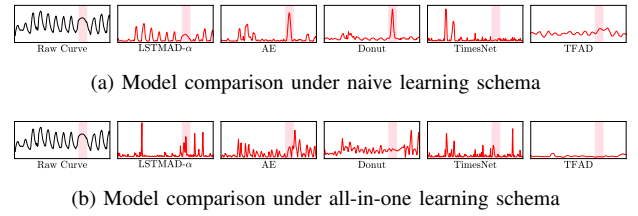(b) Model comparison under all-in-one learning schema

Fig. 12: Case study on CIMIS44AirTemperature2 which contains an easily understandable pattern-wise anomaly.

all results are shown in Table II. Compared to the results without delay-constraint, it can be seen that the performance of some methods significantly deteriorates in this setup, even approaching a 50% decrease (e.g., Anomaly Transformer on TODS). Although EncDecAD is inferior to FITS under all-in-one schema in Fig. 7, when considering delay-constraint, EncDecAD outperforms FITS on TODS, suggesting that some methods may have greater potential in terms of early detection capabilities.

D. Trade-off Between Performance, Cost, and Efficiency

When deploying algorithms in real-world scenarios, it is often necessary to strike a balance between performance, storage costs, and inference speed. Notably different from other surveys, in this benchmark, we prefer inference time over training time due to the practical needs of anomaly detection. Each factor plays a critical role in determining the feasibility and practicality of the deployed algorithm. The trade-offs between these considerations are important to ensure an efficient and effective deployment that meets the specific requirements and constraints of the application. For example, real-time monitoring or critical systems may prioritize efficiency to detect anomalies promptly and respond quickly, while IoT (Internet of Things) devices with limited on-chip memory have to focus on the performance of models with a small number of parameters. We exclude the statistical methods, as their inference time is not a fixed value (polynomially correlated with the volume of historical data). Also, we exclude TFAD because it does not align with the real-time manner. As shown in Fig. 13, the inference time for a single sample is far less than 50 milliseconds for all methods under our experimental platform. The learning-based AR exhibits the most favorable gain-to-cost ratio among all methods. FCVAE takes the longest time to inference due to its multi-epoch MCMC process. EncDecAD has the second longest inference time because it utilizes LSTM to perform inference for 100 time steps. In addition to providing guidance for practical applications, we aim for this perspective and toolkit to assist in discovering the scaling law in the field of time series anomaly detection under more reasonable inductive bias and larger parameter spaces (**RI 14**).

VI. TIMESERIESBENCH TOOLKIT

One of our primary intentions is to develop a suite that liberates practitioners from burdensome workflows, allowing

TABLE II: Performance comparison using all-in-one learning schema under K-delay reduced-length $F1_{best}$ and $AUPRC$

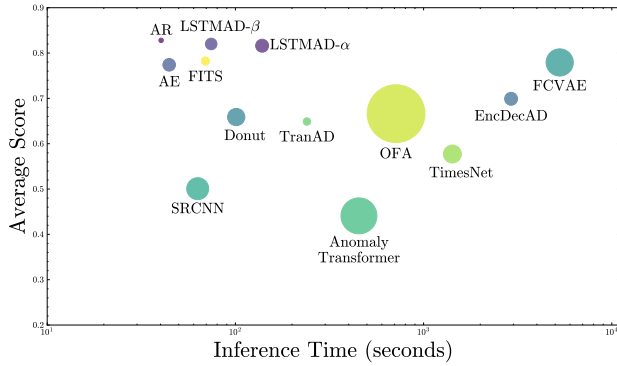| Method	AIOPS (K=10)		NAB (K=150)		TODS (K=3)		WSD (K=10)		Yahoo (K=3)		UCR (K=50)		NEK (K=10)	
	$F1_{best}$	$AUPRC$	$F1_{best}$	$AUPRC$	$F1_{best}$	$AUPRC$	$F1_{best}$	$AUPRC$	$F1_{best}$	$AUPRC$	$F1_{best}$	$AUPRC$	$F1_{best}$	$AUPRC$
MatrixProfile	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SAND	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Sub-LOF	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AR	0.7815	0.766	**0.5334**	**0.407**	**0.6748**	**0.6827**	0.7251	0.6888	**0.9451**	**0.9345**	0.3422	0.3121	0.9137	0.9154
LSTMAD-α	**0.7819**	**0.7924**	0.4235	0.2928	**0.6437**	**0.6424**	**0.7627**	**0.7276**	0.7946	0.7767	**0.4494**	**0.4063**	0.9196	0.9308
LSTMAD-β	**0.7826**	**0.7957**	0.4275	0.2928	0.6234	0.63	**0.7593**	**0.7248**	0.8033	0.7841	**0.4464**	**0.4073**	**0.9458**	**0.9561**
AE	0.7077	0.71	0.3685	0.255	0.5843	0.5551	0.7522	0.719	0.8076	0.7882	0.2233	0.1881	**0.9439**	**0.9513**
EncDecAD	0.6744	0.6434	0.394	0.2602	0.5346	0.5328	0.6641	0.6271	0.5751	0.5277	0.2064	0.1723	0.9348	0.9256
Donut	0.5618	0.5262	0.4014	0.2859	0.6296	0.617	0.6317	0.5765	0.6423	0.6089	0.1869	0.1556	0.5618	0.4383
FCVAE	0.7389	0.7256	**0.5991**	**0.4845**	0.6281	0.6117	0.6838	0.6372	0.847	0.8242	0.3249	0.2813	0.7439	0.6557
SRCNN	0.6253	0.5748	0.3072	0.2383	0.1816	0.1069	0.6267	0.5744	0.1605	0.0978	0.2358	0.2019	0.5446	0.5189
AnomalyTransformer	0.306	0.1596	0.3655	0.2657	0.1331	0.0674	0.4326	0.2876	0.1751	0.0978	0.1243	0.0948	0.6075	0.4891
TFAD	0.4578	0.362	0.15	0.0849	0.4658	0.3988	0.6165	0.569	**0.9281**	**0.9144**	0.4206	0.4034	0.3133	0.1913
TranAD	0.6302	0.5726	0.4406	0.3254	0.4199	0.3944	0.588	0.5219	0.511	0.4551	0.1472	0.1261	0.8932	0.8696
TimesNet	0.4663	0.4199	0.433	0.3105	0.3792	0.3232	0.4063	0.3349	0.5242	0.4988	0.0704	0.0501	0.9028	0.9174
OFA	0.4689	0.3818	0.4145	0.2901	0.4664	0.4147	0.605	0.5593	0.7267	0.6979	0.1718	0.1371	0.7155	0.6536
FITS	0.6623	0.6474	0.4341	0.302	0.5232	0.5423	0.6848	0.6573	0.9013	0.8903	0.2802	0.2585	0.8335	0.8122

Fig. 13: The triad of the model's performance, cost, and efficiency trade-off. The x-axis represents the total inference time (batch size set to 1) of the detectors on AIOPS_7103fa0f-cac4-314f-addc-866190247439 (around 140,000 samples) under default parameters. The y-axis represents the average performance of the models on all datasets under the all-in-one learning schema. The size of the scatter points denotes each method's cube root of the number of parameters.

Despite the built-in settings, TimeSeriesBench toolkit also takes the requirements for extensibility into consideration. We provide flexible interfaces for dataset, method, evaluation criteria, runtime statistic (RT), and learning schema, while existing benchmarks only provides only parts of these interfaces. The dataset interface and algorithm interfaces are provided to allow evaluations on new or private datasets and novel methods under all learning schemas. All plots of raw curves and scores are saved for specific investigation. Previously mentioned flaws in evaluation criteria have sparked a research fervor and several latest works are dedicated to providing evaluation criteria according to different assumptions [30], [32], [34]–[36], [55]. Thus we also expose an interface for swiftly developing neo-criteria for evaluation grounded in realistic assumptions and evaluating the performance of all methods w.r.t various datasets. In some scenarios, there might be a trade-off between model accuracy and model training/testing cost. Therefore, we also provide a runtime statistics interface to conveniently track and analyze the runtime information of the model. Also, you can design your own training, test and other dataflows via implementing learning schema interface.

them to engage in TSAD tasks with greater ease and efficiency. Existing suites narrowly offer convenience for the implementation of new methods with primitive workflows and rudimentary evaluations. Here, benefiting from the modular architecture of TimeSeriesBench, we provide a diverse range of extension interfaces implemented in Python to enable the community to conduct comparative experiments more flexibly and effortlessly, meanwhile reserving the possibilities for innovative and exceptionally demanding experiments, which might unlock unexplored avenues of research.

The overview of the toolkit is shown in Fig. 14. Compared to existing benchmark suites on univariate time series anomaly detection [13]–[15], [54], our framework provides more possibilities for diverse experiment setups. As shown in Table III, in addition to including rich datasets and diverse learning schemas, the toolkit considers the need for more kinds of evaluation criteria. We provide more famous evaluation criteria as built-in criteria. Thus researchers can conveniently compare the strengths and weaknesses of different criteria.
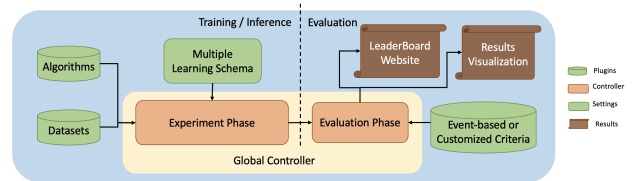

Fig. 14: Overview of TimeSeriesBench Toolkit.

Owing to the flexible framework, our benchmark suite can incorporate real-world scenarios and challenges, aiming to replicate the complexities and nuances encountered in practical applications. This enables researchers and practitioners to gain deeper insights into the strengths and weaknesses of various deep learning approaches, and is well-prepared for the emergence and development of Foundation Models (FMs) in the field of time series anomaly detection. In the toolkit, we provide four kinds of workflows to meet different research needs for time series anomaly detection: algorithms benchmarking, algorithm development, evaluation criteria de-

velopment, and performance analysis. For more information about TimeSeriesBench, please refer to our repository and the website of the leaderboard.

TABLE III: Comparison among TimeSeriesBench and existing UTS anomaly detection benchmark suites. TimeSeriesBench presents more angles for performance evaluation, meanwhile offering user-friendly interfaces for up-to-date model and evaluation criterion developments.

Suites		Exathlon [14]	TODS [13]	TSB [15]	TimeEval [54]	Ours
Data Source	Real-World	✓	✓	✓	✓	✓
	Synthetic	✗	✓	✓	✓	✓
Learning Schema	One-by-one	✓	✓	✓	✓	✓
	All-in-one	✗	✗	✗	✗	✓
	Zero-shot	✗	✗	✗	✗	✓
Supported Eval criteria	Point-based[4]	✓	✓	✓	✓	✓
	Range-based[5]	✓	✗	✓	✓	✓
	Event-based	✗	✗	✗	✗	✓
Extensibility	Dataset	✗	✓	✓	✓	✓
	Method	✓	✓	✗	✓	✓
	Eval criteria	✗	✗	✗	✓	✓
	RT Statistics	✗	✗	✗	✗	✓
	Learning Schema	✗	✗	✗	✗	✓

VII. THREATS TO VALIDITY

As a benchmark, our research is susceptible to several common threats that can compromise the validity and reliability of our findings, including dataset, algorithm settings and the applicable scenarios.

Dataset. Although we have adopted six well-known datasets and have released a real-world dataset, the scenarios covered by these datasets are still limited, so the evaluation results in other scenarios may vary from the main results. We will supplement more datasets in the future to make the evaluation results more generalizable.

Algorithms settings. Due to time and computational resource constraints, we evaluated the algorithms using their default parameter settings in the experiments. If the hyperparameters are fine-tuned for each scenario, some methods might achieve better performance.

Applicable scenarios. In this paper we focus on *real-time* anomaly detection scenarios, and introduce a more robust criterion for evaluation based on real-time detection demands. All results and analysis are based on this setting. However, for offline detection tasks that do not have real-time requirements and require precise detection of the duration of anomalies, other evaluation criteria such as VUS [34] may be more suitable. You can specify this criterion as the evaluation criteria in TimeSeriesBench toolkit to obtain results that are more tailored to your specific application scenario.

VIII. CONCLUSION AND FUTURE DIRECTIONS

In this paper, we propose TimeSeriesBench, a comprehensive and application-oriented benchmark for evaluating the performance of existing and emerging UTS anomaly detection methods. TimeSeriesBench takes into account existing industrial concerns and conducts a comprehensive performance evaluation of some well-known and latest methods under settings that meet industrial requirements. Also, it offers unprecedented perspectives for measuring algorithm performance, meanwhile laying a solid foundation for the development of

Foundation Models in the field. Moreover, TimeSeriesBench includes a user-friendly toolkit and leaderboard, offering easy-to-use interfaces that allow researchers and practitioners to focus on advancing their algorithms without getting entangled in repetitive tasks. We intend to incorporate more latest methods/evaluation criteria and employ more high-quality data. Also, we will continue to monitor the latest developments in general foundation models for time series within the realm of anomaly detection, adapting them accordingly within our benchmark. Additionally, whether you propose deep learning or statistical methods, we welcome your participation in our leaderboard to help drive the development of the time series anomaly detection community.

ACKNOWLEDGMENTS

REFERENCES

[1] 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*, pp. 2828–2837, 2019.

[2] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019.

[3] J. Xu, H. Wu, J. Wang, and M. Long, "Anomaly transformer: Time series anomaly detection with association discrepancy," in *International Conference on Learning Representations*, 2022.

[4] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, *et al.*, "Unsupervised anomaly detection via variational autoencoder for seasonal kpis in web applications," in *Proceedings of the 2018 world wide web conference*, pp. 187–196, 2018.

[5] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017.

[6] Q. Chen, A. Zhang, T. Huang, Q. He, and Y. Song, "Imbalanced dataset-based echo state networks for anomaly detection," *Neural Computing and Applications*, vol. 32, pp. 3685–3694, 2020.

[7] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 387–395, 2018.

[8] Z. Li, W. Chen, and D. Pei, "Robust and unsupervised kpi anomaly detection based on conditional variational autoencoder," in *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, pp. 1–9, IEEE, 2018.

[9] H. Si, C. Pei, Z. Li, Y. Zhao, J. Li, H. Zhang, Z. Diao, J. Li, G. Xie, and D. Pei, "Beyond sharing: Conflict-aware multivariate time series anomaly detection," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, (New York, NY, USA), p. 1635–1645, Association for Computing Machinery, 2023.

[10] Z. Chen, J. Liu, Y. Su, H. Zhang, X. Ling, Y. Yang, and M. R. Lyu, "Adaptive performance anomaly detection for online service systems via pattern sketching," in *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, (New York, NY, USA), p. 61–72, Association for Computing Machinery, 2022.

[11] A. F. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong, "Systematic construction of anomaly detection benchmarks from real data," in *Proceedings of the ACM SIGKDD workshop on outlier detection and description*, pp. 16–21, 2013.

[12] N. Tatbul, T. J. Lee, S. Zdonik, M. Alam, and J. Gottschlich, "Precision and recall for time series," *Advances in neural information processing systems*, vol. 31, 2018.

[13] K.-H. Lai, D. Zha, J. Xu, Y. Zhao, G. Wang, and X. Hu, "Revisiting time series outlier detection: Definitions and benchmarks," in *Thirty-fifth conference on neural information processing systems datasets and benchmarks track (round 1)*, 2021.

[14] V. Jacob, F. Song, A. Stiegler, B. Rad, Y. Diao, and N. Tatbul, "Exathlon: A benchmark for explainable anomaly detection over time series," *arXiv preprint arXiv:2010.05073*, 2020.

[15] J. Paparrizos, Y. Kang, P. Boniol, R. S. Tsay, T. Palpanas, and M. J. Franklin, "Tsb-uad: an end-to-end benchmark suite for univariate time-series anomaly detection," *Proceedings of the VLDB Endowment*, vol. 15, no. 8, pp. 1697–1711, 2022.

[16] S. Schmidl, P. Wenig, and T. Papenbrock, "Anomaly detection in time series: a comprehensive evaluation," *Proceedings of the VLDB Endowment*, vol. 15, no. 9, pp. 1779–1797, 2022.

[17] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *International Conference on Learning Representations*, 2019.

[18] R. Wu and E. Keogh, "Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[19] Y. Research, "A benchmark dataset for time series anomaly detection." https://yahooresearch.tumblr.com/post/114590420346/a-benchmark-dataset-for-time-series-anomaly, 2015.

[20] R. Wu, "Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress." https://wu.renjie.im/research/anomaly-benchmarks-are-flawed/#ucr-time-series-anomaly-archive, 2022.

[21] A. Competition, "Kpi dataset." https://github.com/NetManAIOps/KPI-Anomaly-Detection, 2018.

[22] S. Zhang, Z. Zhong, D. Li, Q. Fan, Y. Sun, M. Zhu, Y. Zhang, D. Pei, J. Sun, Y. Liu, *et al.*, "Efficient kpi anomaly detection through transfer learning for large-scale web services," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 8, pp. 2440–2455, 2022.

[23] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017.

[24] Anonymous, "TEMPO: Prompt-based generative pre-trained transformer for time series forecasting," in *The Twelfth International Conference on Learning Representations*, 2024.

[25] Anonymous, "Time-LLM: Time series forecasting by reprogramming large language models," in *The Twelfth International Conference on Learning Representations*, 2024.

[26] A. Garza and M. Mergenthaler-Canseco, "Timegpt-1," 2023.

[27] T. Zhou, P. Niu, X. Wang, L. Sun, and R. Jin, "One fits all: Power general time series analysis by pretrained LM," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[28] C. Zhang, T. Zhou, Q. Wen, and L. Sun, "Tfad: A decomposition time series anomaly detection architecture with time-frequency analysis," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 2497–2507, 2022.

[29] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang, "Time-series anomaly detection service at microsoft," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 3009–3017, 2019.

[30] K. Doshi, S. Abudalou, and Y. Yilmaz, "Reward once, penalize once: Rectifying time series anomaly detection," in *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2022.

[31] M. E. A. Sehili and Z. Zhang, "Multivariate time series anomaly detection: Fancy algorithms and flawed evaluation methodology," *arXiv preprint arXiv:2308.13068*, 2023.

[32] S. Kim, K. Choi, H.-S. Choi, B. Lee, and S. Yoon, "Towards a rigorous evaluation of time-series anomaly detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 7194–7201, 2022.

[33] A. Garg, W. Zhang, J. Samaran, R. Savitha, and C.-S. Foo, "An evaluation of anomaly detection and diagnosis in multivariate time series," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 6, pp. 2508–2517, 2021.

[34] J. Paparrizos, P. Boniol, T. Palpanas, R. S. Tsay, A. Elmore, and M. J. Franklin, "Volume under the surface: a new accuracy evaluation measure for time-series anomaly detection," *Proceedings of the VLDB Endowment*, vol. 15, no. 11, pp. 2774–2787, 2022.

[35] A. Huet, J. M. Navarro, and D. Rossi, "Local evaluation of time series anomaly detection algorithms," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 635–645, 2022.

[36] E. Scharwächter and E. Müller, "Statistical evaluation of anomaly detectors for sequences," *arXiv preprint arXiv:2008.05788*, 2020.

[37] S. Qin, L. Chen, Y. Luo, and G. Tao, "Multi-view graph contrastive learning for multivariate time series anomaly detection in iot," *IEEE Internet of Things Journal*, 2023.

[38] L. Shen, Z. Li, and J. Kwok, "Timeseries anomaly detection using temporal hierarchical one-class network," *Advances in Neural Information Processing Systems*, vol. 33, pp. 13016–13026, 2020.

[39] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "Usad: Unsupervised anomaly detection on multivariate time series," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 3395–3404, 2020.

[40] Z. Li, Y. Zhao, J. Han, Y. Su, R. Jiao, X. Wen, and D. Pei, "Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 3220–3230, 2021.

[41] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, (New York, NY, USA), p. 93–104, Association for Computing Machinery, 2000.

[42] P. Boniol, J. Paparrizos, T. Palpanas, and M. J. Franklin, "Sand: streaming subsequence anomaly detection," *Proc. VLDB Endow.*, vol. 14, p. 1717–1729, jun 2021.

[43] Y. Zhu, C.-C. M. Yeh, Z. Zimmerman, K. Kamgar, and E. Keogh, "Matrix profile xi: Scrimp++: Time series motif discovery at interactive speeds," in *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 837–846, 2018.

[44] P. J. Rousseeuw and A. M. Leroy, *Robust regression and outlier detection*. John wiley & sons, 2005.

[45] P. Malhotra, L. Vig, G. Shroff, P. Agarwal, *et al.*, "Long short term memory networks for anomaly detection in time series.," in *Esann*, vol. 2015, p. 89, 2015.

[46] A. Ng *et al.*, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[47] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint arXiv:1607.00148*, 2016.

[48] S. Tuli, G. Casale, and N. R. Jennings, "TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data," *Proceedings of VLDB*, vol. 15, no. 6, pp. 1201–1214, 2022.

[49] Z. Wang, C. Pei, M. Ma, X. Wang, Z. Li, D. Pei, S. Rajmohan, D. Zhang, Q. Lin, H. Zhang, J. Li, and G. Xie, "Revisiting vae for unsupervised time series anomaly detection: A frequency perspective," 2024.

[50] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, "Timesnet: Temporal 2d-variation modeling for general time series analysis," in *International Conference on Learning Representations*, 2023.

[51] T. Zhou, P. Niu, X. Wang, L. Sun, and R. Jin, "One fits all: Power general time series analysis by pretrained LM," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[52] Anonymous, "FITS: Modeling time series with \$10k\$ parameters," in *The Twelfth International Conference on Learning Representations*, 2024.

[53] R. Wu, "Ucr_anomalydatasets.pptx, supplemental material to the ucr anomaly archive.." https://wu.renjie.im/research/anomaly-benchmarks-are-flawed/#ucr-time-series-anomaly-archive, 2022.

[54] P. Wenig, S. Schmidl, and T. Papenbrock, "Timeeval: A benchmarking toolkit for time series anomaly detection algorithms," *Proceedings of the VLDB Endowment*, vol. 15, no. 12, pp. 3678–3681, 2022.

[55] D. Wagner, T. Michels, F. C. Schulz, A. Nair, M. Rudolph, and M. Kloft, "Timesead: Benchmarking deep multivariate time-series anomaly detection," *Transactions on Machine Learning Research*, 2023.