EXPERIMENTAL SETTINGS

The machine used in the experiments has two AMD7742 64-core CPUs with 1024GB memory.

Subsequences in each time series are preprocessed with z-score normalization. All final scores output by detectors are normalized in $[0, 1]$. For detectors that rely on randomization, we report the average result of 10 trials on each time series.

[**Datasets**] Synthetic datasets noisy_sine and ARMA are originated from a previous work [9]. Real-world datasets include MIT-BIH Supraventricular Arrhythmia Database (MBA) [7, 13] and other datasets from various domains have been studied in earlier works [9, 10, 23].

Some datasets have two versions, e.g., ann_gun and stdb_308; and each version uses one of the two variables. When either version produces similar AUC for most detectors, we have chosen to use one only. Some datasets are trivial, e.g., chfdb_chf0175 and qtdbsel102; and all detectors have the perfect result (AUC=1), so we do not show them in Table 7.

We labelled anomalous periods for each time series following the previous work [9, 10, 23]. Details are given in Table 9. Positions of anomalies in MBA datasets can be seen in folder "MBA_Annotation".

The period of some datasets varies slightly at different time steps in the series; but it has no effect on the detection accuracy of all algorithms. Our algorithm works well when the subsequence length is set to be roughly the length of the period.

Brief descriptions of some datasets are given as follows.

dutch_pwrdemand: This time series has power consumption for a Dutch research facility for the year 1997 (one power measurement every 15 minutes for 365 days). It shows a characteristic weekly pattern that consists of 5 power usage peaks corresponding to the 5 weekdays followed by 2 days of low power usage on the weekends. Anomalous weeks occur when one or more of the normal usage peaks during a week do not occur due to holidays [9]. There are a total of 672 points each week (672=7x24x60/15) so the period length is 672. The series starts on Wednesday, January 1st, so each week period starts on Wednesday. There are a total of 6 anomalous weeks. Some papers [1, 9, 10] use this dataset with fewer anomalous weeks because they treat continuous anomalous weeks as one anomaly. Additional information about this dataset is given in [31].

ann_gun: It has only one anomalous period when it was first used in Keogh's work [10], as shown in Figure 4. Other anomalous periods in this dataset were later identified [1], and they are shown in Figure 5.

Patient_respiration: Like the previous work [9], we use the subset that begins at 15500 and ends at 22000 from the nprs44 dataset [10]. There are one apparent anomaly and one subtle anomaly in this dataset as shown in Figure 6.

TEK: Following the previous work [9], we also concatenate dataset TEK14, TEK16 and TEK17 as TEK of length 15000. In Keogh's work [10], a total of 4 anomalies are marked. But TEK14 has 2 anomalous snippets belonging to the same period as shown in Figure 7. Since we regard each anomaly as an anomalous subsequence of one complete period, it is treated as one anomalous periodic subsequence of length 1000. So there are a total of 3 anomalous subsequences in our annotations of this dataset.

Table 9: Locations of anomalous periodic subsequences in each dataset in terms of index i in $Y_{i,m}$, where the period length is m. For the last five datasets, see folder: MBA_Annotation at https://anonymous.4open.science/r/AnomalyDetectionTS-04A2/.

Dataset	period length	anomalous period index i
noisy_sine	300	6,11,21,31
ARMA	500	101,102,161
GPS_trajectory	2200	3,6
Patient_respiration	150	7,34
TEK	1000	2,10,13
dutch_pwrdemand	672	1,13,18,19,20,52
ann_gun	150	3,15,16,17,19
mitdb_100_180	250	8
mitdbx_108	370	12,28,29,30,31
stdb_308	400	7
lstdb_20221_43	170	5
lstdb_20321_40	200	5
MBA803	105	
MBA805	100	see details
MBA806	75	in folder:
MBA820	100	MBA_Annotation
MBA14046	90	

MBA803,MBA805,MBA806,MBA820,MBA14046: These datasets are subsets of the full MBA dataset, as used in the previous work [1].

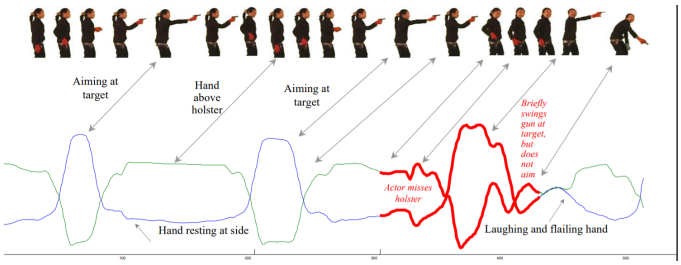

Figure 4: One anomaly period in the ann_gun dataset. The diagram is extracted from [10].

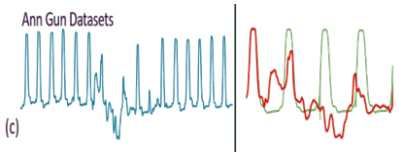

Figure 5: Additional anomalous periods in the ann_gun dataset, as identified by [1]. The diagram is extracted from [1].

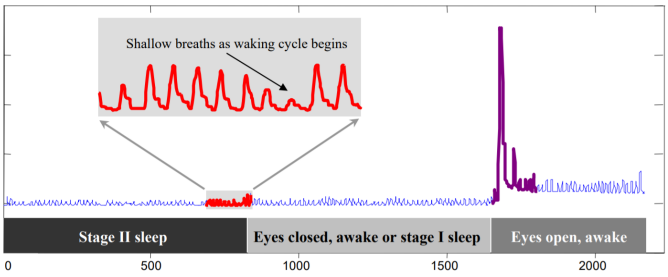

Figure 6: Anomalies in Patient_respiration dataset. The diagram is extracted from [10].

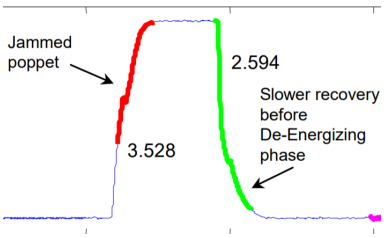

Figure 7: Anomalies in a period of TEK14 dataset. The diagram is extracted from [10].

[**Algorithms**] The STOMP [33] implementation of MP is used; NormA is from http://helios.mi.parisdescartes.fr/ themisp/norma/; IDK-based detectors are our implementations based on [28]; and WFD is from github.com/GAMES-UChile/Wasserstein-Fourier. Others are from scikit-learn.org. All are in Python.

As for the 1Line method, we use one of the following five types of basic vectorized primitive functions in Matlab as an anomaly score for each sliding window of size ω:

(i) $\pm$diff(Y): the difference between the current point and the previous point. Here $\omega = 1$.
(ii) $\pm$movmax(Y, $\boldsymbol{\omega}$)
(iii) $\pm$movmin(Y, $\boldsymbol{\omega}$)
(iv) $\pm$movmean(Y, $\boldsymbol{\omega}$)
(v) $\pm$movstd(Y, $\boldsymbol{\omega}$)

where Y is the time series; and the maximum, minimum, mean or standard deviation is computed for each window of $\boldsymbol{\omega}$ points.

We run these 5 one-liner on each dataset and report the median AUC (out of the five values) in Table 7. Low median values indicate that the datasets are hard to detect using the 1Line method; otherwise, the datasets have anomalies that can be easily detected.

Figure 8 shows the Friedman significance test result for the five top-ranked distribution-based and the three sliding-window-based detectors in the experiment.

[**Measures**] The detection accuracy of an anomaly detector is measured in terms of AUC (Area under ROC curve). As all the anomaly detectors are unsupervised learners, all models are trained with the given datasets with no labels. Only after the trained models have made predictions, the ground truth labels are used to compute the AUC for each dataset.

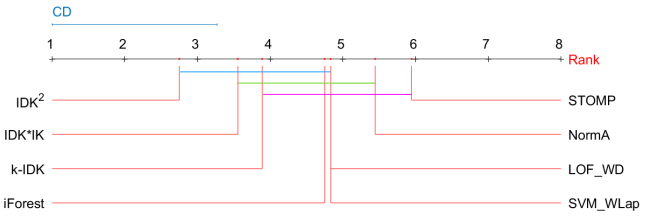

Figure 8: Friedman-Nemenyi test for the five top-ranked distribution-based and the three sliding window-based detectors at significance level 0.1. If two algorithms are connected by a CD (critical difference) line, then there is no significant difference between them.

Given a periodic time series Y of length n and period length m, a subsequence $Y_{i,m}$ of Y is a subset of contiguous values of length m, for $i = 1, \ldots, s$, where $s = \lfloor n/m \rfloor$. A distribution-based (non-sliding-window) algorithm outputs a score of each periodic subsequence $Y_{i,m}$. Then AUC can be calculated based on scores α_i for $Y_{i,m}$ $\forall i = 1, \ldots, s$.

An anomaly detector using the sliding window size ω produces a total of $n - \omega + 1$ subsequences from Y. When calculating AUC, scores of the sliding subsequences are transformed into periodic subsequence scores as follows: Let S_j be the anomaly score of subsequence $Y_{j,\omega}$, where $1 \leq j \leq (n - \omega + 1)$. The final score corresponds to a periodic subsequence $Y_{i,m}$ is the maximum score of S_j $\forall j$ such that at least half of $Y_{j,\omega}$ is included in $Y_{i,m}$.

8 DISCUSSION

It is interesting to note that NormA is motivated to tackle noise by using a summarized normal model [1]. The paper has verified that NormA could detect anomalies in a time series corrupted uniformly with Gaussian noise of the same level. However, we showed that, on the noisy_sine dataset where normal and anomalous subsequences have different noise levels, NormA failed to detect the anomalies.

A recent paper has questioned the claims made by deep learning models because many time series datasets used can be solved with similarly high accuracy by using a single line of standard library Matlab code [32]. A different study also found that a deep learning model performed significantly poorer than NormA, even it has been given an unfair advantage of training using normal subsequences only (see [1] for details.)

We have shown that the proposed distributional treatment works well for periodic time series. It would be interesting to examine how well it could deal with aperiodic time series in the future. Our work also opens up opportunities to use the same treatment for other data mining tasks in time series. The key to all these endeavors is to define what a subsequence is, to be represented as an iid sample of an unknown pdf.

It is interesting to note that Isolation Forest (iForest) [12] with its default setting performed better than or competitive to many more complicated time series anomaly detectors that require a mixture of different algorithms/methods (see Table IV in [1] for details.) In the context of point anomaly detection, iForest has been shown to be closely related to a kernel-based anomaly detector based on IDK;

but iForest is weaker than IDK because of its isolation mechanism and it has no distributional characterization (see Section 7 in [27] for details.)

IDK*IK is equivalent to using an isolation-based point anomaly detector such as Isolation Forest [12] and iNNE after the given dataset has been mapped to level-1 Hilbert space. A close relationship between isolation-based anomaly detection and IDK anomaly detector has been revealed recently (see Section 7 in [27].)

Our result in Table 7 shows that OCSVM working in the time domain (WLap) may be slightly better than in frequency domain (WFLap); but the difference is small. A recent work shows that WFD is better than Euclidean distance in kNN and logistic classifications [3]. However, they did not compare time versus frequency domains using the same Wasserstein distance like we did.

Conceptually, WD should be as good as IDK. Our result uncovers weaknesses in WD which could possibly occur in optimizing a transportation plan and the granularity of the histogram representation. IDK, being a non-parametric method, has no such issues.

A recent study on time series measures [17] have included 4 embedding measures which employ a similarity measure to construct new representations such that the Euclidean distance of the two representations approximately preserves the similarity of the corresponding original time series as measured by the similarity measure used to construct the representations. Note that these embedding measures differ conceptually from distributional measures such as WD, KME and IDK, which are more fundamental measures, that they may use in their embedding. While they show promise, they are at least one order slower than Euclidean distance. In addition, the study [17] is conducted using 1-NN classifier only.