Parameter-free Streaming Distance-based Outlier Detection

Apostolos Giannoulidis
Aristotle University of Thessaloniki
Thessaloniki, Greece
agiannous@csd.auth.gr

Nikodimos Nikolaidis
Atlantis Engineering
Thessaloniki, Greece
nikolaidis@atlantis-engineering.com

Anastasios Gounaris
Aristotle University of Thessaloniki
Thessaloniki, Greece
gounaria@csd.auth.gr

***Abstract*—Dealing with anomaly detection in streaming data using distance-based techniques suffers from sensitivity to the input parameters. To solve this problem, we propose a parameter-free self-tuning solution, which can decide optimized parameters on the fly. Our technique achieves better or equal performance with traditional distance-based solutions even when these have been fine-tuned.**

***Index Terms*—distance-based outlier detection, data streams**

I. Introduction

Distance-based outlier detection is a particularly popular family of techniques for data anomalies [1]–[3] and one of its strongest advantages is that lends itself to efficient implementations for data streams [4]–[15]. Informally, it employs two main parameters, k and R along with a distance function. Any object (a.k.a. data point) for which less than k other data points are within distance R is considered an outlier.

These two values, namely k and R, are the two main input parameters and the performance of distance-based techniques is highly dependent on them, given also the curse of dimensionality, which is existent in multidimensional settings, like multivariate time series. Running multiple combinations of k and R, e.g., [14], [16] does not solve the problem of auto-configuring the parameters despite the performance penalty it incurs. Moreover, any self-tuning solutions, to be applicable for data streams in an online manner, need to require no training and usage of historical data. To deal with the aforementioned challenges, we propose a parameter-free self-tuning distance-based outlier detection method that can outperform distance-based techniques even after the latter are fine-tuned.

Related Work. A review on autoML for anomaly detection is conducted in [17], where, in the case of unsupervised techniques, it explicitly highlights the need for novel auto-tuning techniques. A recent work on automatic anomaly detection is presented in [1], where two novel methodologies for automated anomaly detection are presented; nevertheless, both of them are not applicable in a streaming setting. Regarding streaming anomaly detection, there are several methodologies in the literature. In [15], tree-based techniques such as HST and RRCF [18], [19], projection-based techniques such LODA and xStream [20], [21], along with proximity and density-based techniques are reviewed and analyzed. Another method for streaming anomaly detection, called SAND, is proposed in [22]. SAND tries to find anomalies in univariate time series by looking into sub-sequences of time series. A common approach for online settings is the use of distance-based outlier detection, where anomalies are detected based on the proximity of data using the concept of time window [23], [24]. In this case, the need for effective parameter tuning is aggravated and thus approaches for considering multiple sets of parameters simultaneously are proposed [14]. Although multiple parameters provide a broader image regarding outliers in the data streams, the user still has to decide the anomaly thresholds. In general, the problem of tuning thresholds in anomaly detection is not new and has been explored in works like [25] and [26].

Contribution and Structure. Our contribution lies in the proposal of a novel technique for automated distance-based outlier detection, inspired by tuning techniques for anomaly score thresholds [25]. The proposed solution (i) aims to auto-tune the selection of the k and R parameters of distance-based outlier detection, (ii) is tailored to data streams, and (iii) requires no training at all. Moreover, the solution inherently adapts to concept drifts, because it continuously refines the parameters and, in our experiments, achieves higher performance against the statically fine-tuned solution of classical distance-based outlier detection. On average, the F1 score when the objective is to detect an anomaly within a range increases by 2.2X. The complete codebase is available.[1]

In Sec. II we introduce the definitions and the methodology, while in Sec. III, we evaluate our technique against a baseline. We conclude in Sec. IV.

II. Definitions and Methods

Definition 1 (Object neighbors [2], [23]): . Let $R > 0$ be a positive threshold. Two data objects p_i and p_j are neighbors, if the distance between them according to a defined function *dist()* is no larger than R. The function $nn(p_i, R)$ denotes the number of neighbors that a data object p_i has, given the parameter R.

This research was carried out as part of the project "PRECognition" (Project code: KMP6-0077768) under the framework of the Action "Investment Plans of Innovation" of the Operational Program "Central Macedonia 2014 2020", that is co-funded by the European Regional Development Fund and Greece.

[1]https://github.com/agiannoul/ADBOD

TABLE I
NOTATION USED IN THIS WORK

Annotation	Description
p_t	A multidimensional data point with timestamp t
w	The size of the (sliding) window
s	The size of the slide
W_i	A set with all the data points in a particular time window
O_i	A set with all the outliers in the W_i
$d_k(p_t)$	The distance of p_t from its k^{th} nearest neighbor inside a window
D_k	A set of $d_k(p_t)$ values for all data points in a window
C	A set of candidate k parameter values
c_i	A particular choice for k parameter from C
R^k	A set of candidate R parameters for a particular k
r^{c_i} or r^k	A particular choice of the R parameter from R^k
$\mu(D_k)$	mean of distances in D_k (mean of anomaly scores)
$\sigma(D_k)$	standard deviation of distances in D_k (standard deviation of anomaly scores)

Definition 2 (Distance-based outlier [2]): Given a set of points P and the threshold parameters R and k, a point $p_i \in P$ is an outlier if the number of its neighbors $nn(p_i) < k$, i.e., the number of points $p_j, j \neq i$ for which $dist(p_i, p_j) \leq R$ is less than k. Equivalently, given a set of points P and the threshold parameters R and k, a point $p_i \in P$ is an outlier if its distance from its k^{th} closest neighbor $d_k(p_i)$ is greater than R.

To apply distance-based outlier detection over time series or streaming data, the *sliding window* concept is employed. In its main count-based form, only the last w values of the time series are taken into account and window slides may not necessarily occur after a new point arrives; the slide length is defined as s. In general, sliding windows partially overlap on the time axis. If $s = w$, we have the specific case of tumbling windows, whereas values of $s > w$ are rarely considered because they essentially discard data points from processing.

Given a multivariate time series $P = \{p_0, p_1, ..., p_n\}$, where each data point p_t is collected at a specific timestamp t with $p_t \in \mathbb{R}^m$ (m is the dimensionality of that data), our task is to report distance-based outliers that correspond to real anomalies in the data. In continuous outlier detection over streams, given a window W of length w and slide of length s, we report distance based outliers for all windows $W_i = \{p_{s*i}, p_{s*i+1}, \ldots, p_{s*i+w-1}\}$ until $w + s * i > n$. This process generalizes to infinite streams and time-based windows in a straightforward manner. Table I summarizes the notation.

A. Distance-based Outlier Detection (DOD) in Data Streams

By using the concept of sliding window and Definition 2, we can report outliers for each window slide. To this end, any exact continuous distance-based outlier can be used [4]–[13], [15]. These solutions differ in their latency but they report the same outliers. In a brute-force manner, to detect outliers in every W_i, the pairwise distances between all data in W_i are calculated. Based on these distances, we can construct the set of outliers $O_i = \{p_i \in W_i | d_k(p_i) > R\}$.

To yield more concise results, post processing across multiple slides is possible following many strategies. For example, we may report an outlier only in the first window slide it appears, or if there is at least one window slide where the corresponding data point is active, or if it is an outlier in all window slides, to which it belongs.

B. Dynamic K and R (Dyn)

Selecting an optimal k and R parameter combination is not straightforward and, moreover, in a streaming environment, it is not realistic to assume that processing historic data, i.e., data from past windows, is a practical approach. In addition, there is no guarantee that a particular choice of parameters will continue to be appropriate in the future since the behavior of data may change due to concept drifts. In our solutions, we tackle these challenges through automatically updating k and R in each window W_i.

Our novel solution is inspired by techniques to dynamically set thresholds on anomaly scores, e.g., in demanding predictive maintenance use cases [25], [27]. We use an objective function to define the R and k parameters in each window to detect distance-based outliers. More specifically, in each window slide, we consider a predefined set of candidate values for parameter k, C containing natural numbers. For each $c_j \in C$,

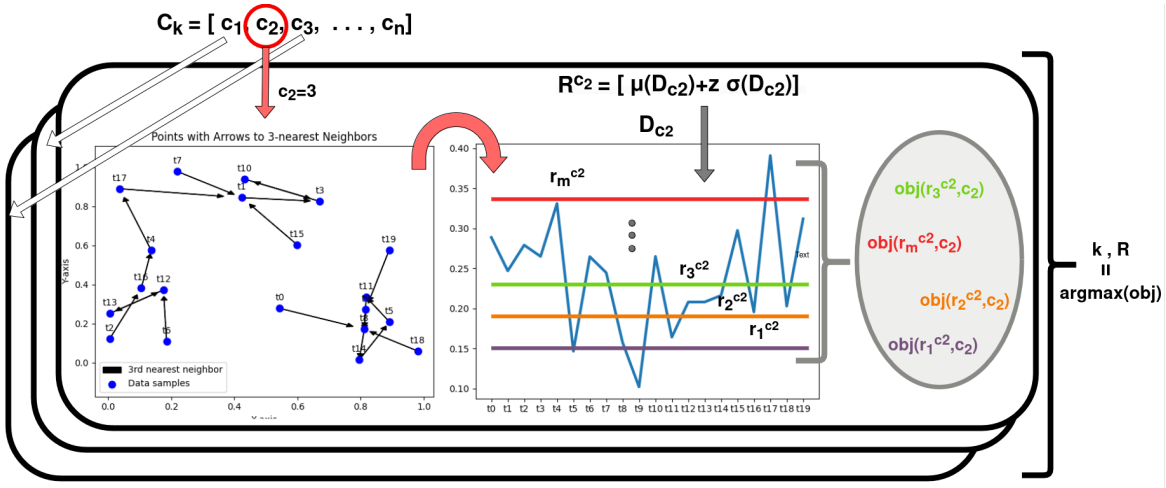

Fig. 1. Overview of our solution for selecting an optimal k and R combination for each window slide.

we calculate a set of possible R values based on the distances from the c_j^{th} nearest neighbor of all data points in the window, notated as $D_k = \{d_k(p_t), \forall p_t \in W_i\}$. The candidate values of R parameters for a specific c_j (or k) parameter are denoted as $R^{c_j} = \mu(D_{c_j}) + z * \sigma(D_{c_j})$, where z belongs to a predefined range. In our experiments $z \in [1, 10]$. Thus, overall $|C| * |z|$ pairs of k and R parameters are examined.

Then, leveraging the idea in [25], we select the parameters according to the following formula, where $r_i^{c_j} \in R^{c_j}$, $i = 1, \ldots, |z|$:$(k, R) = \text{argmax}_{(c_j, r_i^{c_j})}\, obj(c_j, r_i^{c_j})$

$$obj(c_j, r_i^{c_j}) = \frac{\Delta\mu(D_{c_j})/\mu(D_{c_j}) + \Delta\sigma(D_{c_j})/\sigma(D_{c_j})}{|O_i|}$$

where, $\Delta\mu(D_{c_k}) = \mu(D_{c_k}) - \mu(d_k \in D_{c_k} | d_k < r_i^{c_j})$ and $\Delta\sigma(D_{c_k}) = \sigma(D_{c_k}) - \sigma(d_k \in D_{c_k} | d_k < r_i^{c_j})$.

The main idea is to select the combination of parameters k and R that incur the greatest decrease in the mean and standard deviation of distances to the k^{th} nearest neighbor when outliers are removed, while also penalizing large numbers of outliers. The intuition behind this is that such a combination better distinguishes some points (the anomalies) from the remainder of the window contents. An illustration of that process for a specific c_k in a specific slide is depicted in Figure 1.

An important note is that instead of using the raw distances of the k^{th} nearest neighbor D_{c_j} to calculate candidate thresholds, it is preferable to apply a min-max normalization on D_{c_j} since, as k increases, the distances tend to have smaller proportional differences, which would affect the effectiveness of the objective function. Finally, we need to account for the cases in which there are actually no outliers. The objective function inherently considers some of the points as outliers and detects the manner in which these are most distinguishable. This problem was not encountered in dynamic thresholding works, such as [25], due to the use of historic values; we differ in that we update parameters in every single slide. To address the problem of always reporting outliers even if the D_{c_j} distances do not differ significantly, p is reported as an outlier iff $\frac{d_k(p) - R}{d_k(p)} > \epsilon$, where ϵ is a small constant (in our experiments, we set it to 0.05).

III. Evaluation

To evaluate the performance of our *Dyn* proposal, we choose the Yahoo Dataset [28], which contains 4 sets of univariate time series (referred to as A1-A4) with labeled anomalies, summing in total to 367 time series. On average, the length of each time series is 1561 and anomalous points are approximately 0.4% of the total dataset. Since we are interested in multivariate anomaly detection, we consider subsequences of the time series to perform anomaly detection. So for each timestamp, we use the last s_l (sub-sequence length) of points as a single vector. We test W values of 200 and 400, while the slide is half the size of W for all experiments. Furthermore, we test s_l values of 2, 10, and a suggested subsequent length based on auto-correlation, as calculated in [29]. When $s_l > 2$, we also test feature extraction (e.g., mean, standard deviation, and so on) from the sub-sequence instead of analyzing raw data. Finally, we check several values for k and R parameters (42 combinations in total, in the case of DOD). Min-max is initially not employed. Full details and reproducible settings are available in our code repository.

To measure the performance of the techniques, we use the range-based metrics of recall and precision [30]. More specifically we adopt the AD1 and AD2 levels as defined in the Exathlon benchmark [31]. In AD1, we get the maximum recall value, i.e. 1, if an outlier is detected within an anomaly range, while AD2 recall is proportional to the relative size of detected outliers reported during the anomalous periods. In both cases, the precision is computed as the ratio of the outliers detected within the anomaly range and all detected outliers. Due to space limitations, we report only F1 scores.

Comparison against global choices of k and R. Initially, we pick the parameters for which the median of AD1 F1 and AD2 F1 across all time series is maximized. This allows us to evaluate the best choice of k and R parameters for *DOD* against *Dyn*. The results are in Figure 2. It is worth noticing that to achieve the results shown, for *Dyn*, only 12 different combinations of parameters are examined (since no k and R have to be defined), while for *DOD* 504 parameter combinations are tested.

The results provide evidence that using a constant fine-tuned choice of k and R for *DOD* provides worse results than our auto-tuned *Dyn*, i.e., our solution is both effective and easy to employ. The average AD1 F1 increases in the datasets range from 1.65X to 3.44 (mean 2.2X). For AD2, the average increase is 1.24X.

Comparison against locally optimal choices of k and R. We also test the performance of *DOD*, when, instead of selecting globally optimal k and R parameters for all time series in the dataset, we fine-tune the selection of k and R for each time series individually. The results can be seen in Figure 3. *Dyn* is still better in AD1 F1 (1.89X F1 average increase) and achieves similar performance regarding AD2 F1. The main difference in this experiment is that *Dyn* selects an optimal k and R combination for each window, while *DOD* selects a combination that remains constant for the complete time series, which is problematic in large time series because it cannot adapt to drifts.

Finally, we note that the achieved results on the Yahoo dataset are similar to or better than those reported in [29] in absolute numbers, but no direct comparison can be made. This provides evidence that distance-based outliers is a suitable solutions for detecting anomalies in this specific dataset.

Generalization. Extending the experiments above by implementing min-max normalization on the data, we derive the general remark that the *Dyn* approach achieves better or similar results with globally fine-tuned *DOD*, while only in one case (when using normalization), DOD achieves better results. When we fine-tune *DOD* for each time-series, *Dyn* achieves a higher median AD1 F1 score in 41.7% of the cases and in 45.8% of the cases, the results were similar (difference less than 0.02 in F1); in only 12.5% of the cases

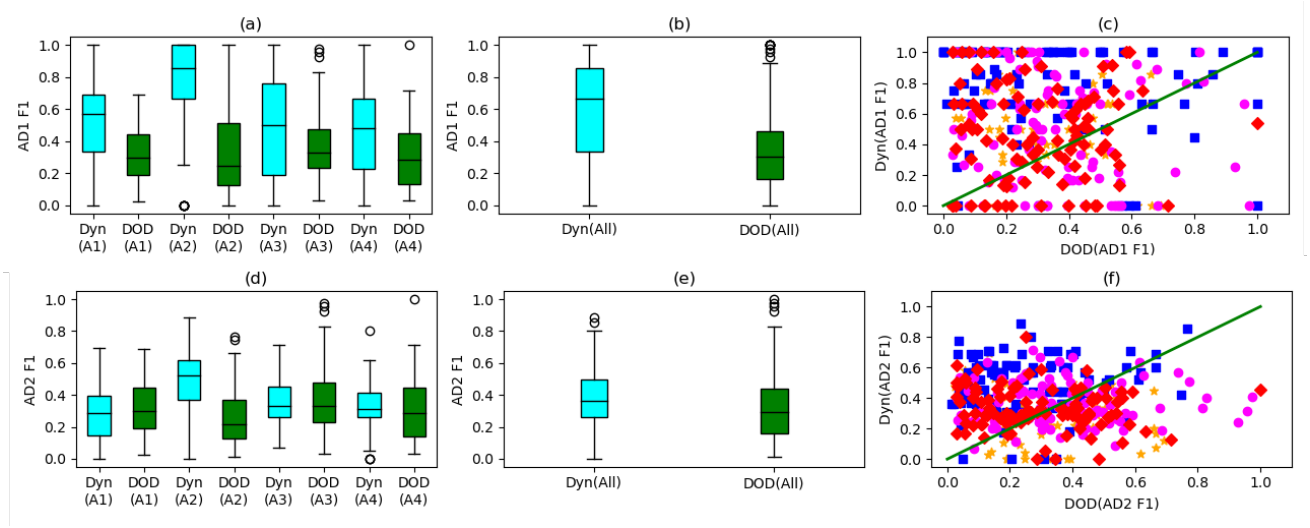

Fig. 2. Results for fine-tuned parameters on Yahoo Dataset produced by a global k and R combination, which achieves the highest median AD1 and AD2 F1 score respectively. Each mark in (c) and (f) represents one time series from Yahoo Dataset and each mark type denotes a different time series.

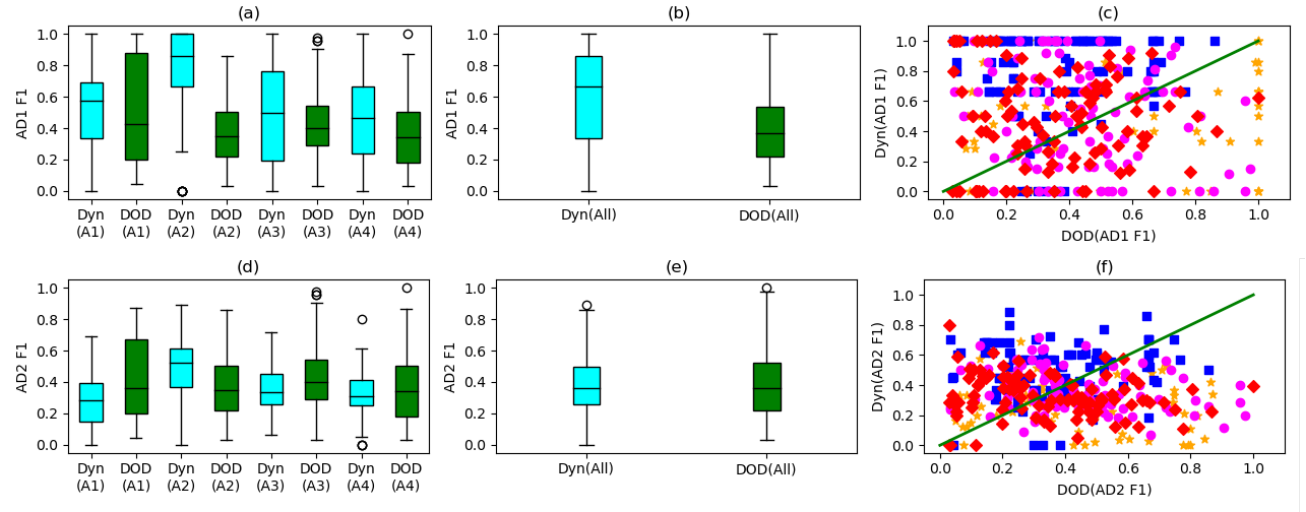

Fig. 3. Results when we select best k and R for DOD for each time series separately.

Dyn performed worse.

Regarding execution time, if we use *DOD* without any optimization (like MCOD [23] or multi-query flavors [14], [16]), both techniques have complexity $O(|W|^2)$. In Figure 4, we present the runtime in all time series, for different W and sub-sequence lengths. All the experiments were conducted on a single machine with 16 GB memory and an Intel core i5 CPU with 4 cores operating at 3.20GHz. For *DOD*, the execution needs to be repeated for each k and R combination investigated, which is not the case for *Dyn*.

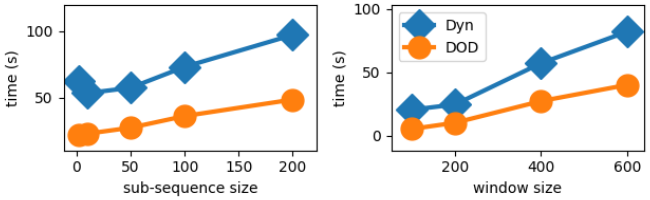

Fig. 4. Execution time for different W and sub-sequence size.

IV. Conclusion and Future Work

We presented a self-configuring distance-based outlier detection technique, which provides better or equal results in most of the cases compared to a fine-tuned traditional distance-based approach. Zero training or initial configuration is required when applied to time series, thus it is particularly useful for streaming scenarios, while it is inherently adaptive to concept drifts. In the future, we plan to extend our solution to increase its efficiency and investigate data engineering issues involved and additional objective functions in depth.

REFERENCES

[1] L. Cao, Y. Yan, Y. Wang, S. Madden, and E. A. Rundensteiner, "Autood: Automatic outlier detection," *Proc. ACM Manag. Data*, vol. 1, no. 1, may 2023. [Online]. Available: https://doi.org/10.1145/3588700

[2] E. M. Knorr, R. T. Ng, and V. Tucakov, "Distance-based outliers: Algorithms and applications," *The VLDB Journal*, vol. 8, no. 3-4, 2000.

[3] A. Boukerche, L. Zheng, and O. Alfandi, "Outlier detection: Methods, models, and classification," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 55:1–55:37, 2020.

[4] F. Angiulli and F. Fassetti, "Detecting distance-based outliers in streams of data," in *CIKM*, 2007, pp. 811–820.

[5] D. Yang, E. Rundensteiner, and M. Ward, "Neighbor-based pattern detection for windows over streaming data," in *EDBT*, 2009, pp. 529–540.

[6] L. Cao, D. Yang, Q. Wang, Y. Yu, J. Wang, and E. A. Rundensteiner, "Scalable distance-based outlier detection over high-volume data streams," in *ICDE*, 2014, pp. 76–87.

[7] M. Kontaki, A. Gounaris, A. N. Papadopoulos, K. Tsichlas, and Y. Manolopoulos, "Efficient and flexible algorithms for monitoring distance-based outliers over data streams," *Information systems*, vol. 55, pp. 37–53, 2016.

[8] L. Cao, J. Wang, and E. A. Rundensteiner, "Sharing-aware outlier analytics over high-volume data streams," in *ICDM*. ACM, 2016, pp. 527–540.

[9] L. Tran, L. Fan, and C. Shahabi, "Distance-based outlier detection in data streams," *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 1089–1100, 2016.

[10] L. Cao, Y. Yan, C. Kuhlman, Q. Wang, E. A. Rundensteiner, and M. Y. Eltabakh, "Multi-tactic distance-based outlier detection," in *ICDE*, 2017, pp. 959–970.

[11] G. Zhao, Y. Yu, P. Song, G. Zhao, and Z. Ji, "A parameter space framework for online outlier detection over high-volume data streams," *IEEE Access*, vol. 6, pp. 38 124–38 136, 2018.

[12] S. Yoon, J. Lee, and B. S. Lee, "NETS: extremely fast outlier detection from a data stream via set-based processing," *PVLDB*, vol. 12, no. 11, pp. 1303–1315, 2019.

[13] L. Tran, L. Fan, and C. Shahabi, "Distance-based outlier detection in data streams," *PVLDB*, vol. 9, no. 12, pp. 1089–1100, 2016.

[14] T. Toliopoulos, C. Bellas, A. Gounaris, and A. Papadopoulos, "Proud: Parallel outlier detection for streams," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 2717–2720. [Online]. Available: https://doi.org/10.1145/3318464.3384688

[15] A. Ntroumpogiannis, M. Giannoulis, N. Myrtakis, V. Christophides, E. Simon, and I. Tsamardinos, "A meta-level analysis of online anomaly detectors," *The VLDB Journal*, pp. 1–42, 2023.

[16] T. Toliopoulos and A. Gounaris, "Multi-parameter streaming outlier detection," in *WI*, 2019, pp. 208–216.

[17] M. Bahri, F. Salutari, A. Putina, and M. Sozio, "Automl: state of the art with a focus on anomaly detection, challenges, and research directions," *International Journal of Data Science and Analytics*, vol. 14, no. 2, pp. 113–126, 2022.

[18] S. C. Tan, K. M. Ting, and T. F. Liu, "Fast anomaly detection for streaming data," in *Twenty-second international joint conference on artificial intelligence*, 2011.

[19] S. Guha, N. Mishra, G. Roy, and O. Schrijvers, "Robust random cut forest based anomaly detection on streams," in *International conference on machine learning*. PMLR, 2016, pp. 2712–2721.

[20] T. Pevnỳ, "Loda: Lightweight on-line detector of anomalies," *Machine Learning*, vol. 102, pp. 275–304, 2016.

[21] E. Manzoor, H. Lamba, and L. Akoglu, "xstream: Outlier detection in feature-evolving data streams," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1963–1972.

[22] P. Boniol, J. Paparrizos, T. Palpanas, and M. J. Franklin, "Sand: Streaming subsequence anomaly detection," *Proc. VLDB Endow.*, vol. 14, no. 10, p. 1717–1729, jun 2021. [Online]. Available: https://doi.org/10.14778/3467861.3467863

[23] M. Kontaki, A. Gounaris, A. N. Papadopoulos, K. Tsichlas, and Y. Manolopoulos, "Efficient and flexible algorithms for monitoring distance-based outliers over data streams," *Information Systems*, vol. 55, pp. 37–53, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306437915001349

[24] T. Toliopoulos, A. Gounaris, K. Tsichlas, A. Papadopoulos, and S. Sampaio, "Continuous outlier mining of streaming data in flink," *Information Systems*, vol. 93, p. 101569, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306437920300594

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

[26] A. Giannoulidis, A. Gounaris, N. Nikolaidis, A. Naskos, and D. Caljouw, "Investigating thresholding techniques in a real predictive maintenance scenario," *ACM SIGKDD Explorations Newsletter*, vol. 24, no. 2, pp. 86–95, 2022.

[27] H. Meng, Y. Zhang, Y. Li, and H. Zhao, "Spacecraft anomaly detection via transformer reconstruction error," in *Proc. of the Int. Conf. on Aerospace System Science and Engineering 2019*, Z. Jing, Ed., 2020, pp. 351–362.

[28] "Yahoo! webscope dataset s5 - a labeled anomaly detection dataset, version 1.0(16m)," https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70.

[29] 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.

[30] 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.

[31] 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.