Time Series Anomaly Detection with Untrained Convolutional Kernels

Wenjie Xi* Jessica Lin*

Abstract

Time series anomaly detection is critical for applications ranging from industrial monitoring to financial fraud detection. Among existing methods, C^{22}MP represents the state-of-the-art by enhancing anomaly detection through the combination of the Matrix Profile and catch22, a handcrafted feature set, to improve anomaly detection performance. However, catch22 features are limited in their ability to capture various characteristics in time series data. In this work, we propose a novel framework, RandomAD, that replaces catch22 with thousands of convolution kernels with random parameters. Additionally, we introduce a kernel selection mechanism to retain only the most informative kernels and a multi-window selection strategy with anomaly filtering module to optimize the choice of window size and final result. Through extensive experiments on the benchmark datasets, we demonstrate that our approach outperforms all tested state-of-the-art methods.

1 Introduction

Time series anomaly detection (TSAD) aims to detect unusual subsequences (often referred to as *discords* when they are the most anomalous) or time points within long data streams. Due to its significance in applications ranging from industrial monitoring to fraud detection [11], it has attracted considerable attention from researchers [19]. One remarkable approach for finding discords is the Matrix Profile (MP) [25], a data structure that stores each subsequence's Euclidean distance to its nearest neighbor. MP-based methods [15, 25] are effective at finding anomalies that stand out due to their distinctive shapes.

However, relying solely on shape-based comparisons has limitations. In many real-world applications, anomalies may not present as distinct shape but rather as subtle variations in statistical or dynamic properties. Incorporating information such as variance, entropy, or autocorrelation can capture these subtle changes, thereby improving the performance. To address this problem, C^{22}MP [21] is introduced as an extension of the traditional MP framework by integrating catch22 [16], a collection of 22 domain-independent time series features. By using catch22 to extract features from each subsequence and with an early-abandoning mechanism, C^{22}MP can detect anomalies efficiently and accurately. While this fusion is able to detect feature-based anomalies and improve TSAD performance, it also brings two challenges since it relies on a fixed set of handcrafted features: (1) it cannot capture complex anomalies outside the predefined feature space, and (2) under semi-supervised setting, it requires manual tuning to select the most relevant features for a specific dataset.

Inspired by the success of random convolutional kernel techniques in time series classification [6, 7, 22] and clustering [13], we propose RandomAD, which extracts a rich set of features using thousands of random convolutional kernels. Specifically, we adopt the random kernel generation mechanism in MiniRocket [7] to generate thousands of different convolutional kernels with randomly initialized weights based on the training set. To improve performance, we propose a kernel selection mechanism to retain only the most informative kernels capable of effectively capturing underlying patterns. Each selected kernel is then applied to extract features from subsequences, transforming each subsequence into a feature representation.

Additionally, we introduce a multi-window selection strategy to adaptively determine the window sizes used in the framework. Existing methods typically rely on fixed window sizes or manually set magic numbers, which are often suboptimal when handling different types of anomalies with varying characteristics [14]. Some approaches are proposed to address such problem. For example, MADRID [14] efficiently computes time series discords across all subsequence lengths by leveraging shared computations, iterative doubling, and forward pruning. Similarly, MERLIN [17] iteratively compares subsequences of varying lengths with their immediate neighbors in a parameter-free manner to detect discords. However, both methods rely on raw distance comparisons between subsequence and still require a large search space for window selection.

In contrast, our approach adopts a large number of random convolutional kernels to extract a diverse set of features from subsequences. These random kernels highlight specific local regions of interest and

*Department of Computer Science, George Mason University, United States. {wxi, jessica}@gmu.edu.

therefore are less sensitive to small changes in window size. This property enables us to restrict the search to a small set of candidate window sizes within a predefined range, rather than exhaustively testing every possible value. We will discuss this in more detail in Section 4.5. Furthermore, by integrating the anomaly filtering mechanism, which selects the most effective window size based on anomaly scores, RandomAD can automatically identify an appropriate window size and produce reliable final anomaly detection results.

In summary, our contributions are as follows:

- To the best of our knowledge, we are the first to introduce random kernel feature extraction for time series anomaly detection and propose a novel kernel selection mechanism to identify effective kernels in a semi-supervised setting.
- We propose a multi-window selection strategy combined with a anomaly filtering mechanism that automatically determines the optimal window size and final anomaly detection result based on anomaly scores.
- Extensive experiments on 250 datasets demonstrate that our proposed method outperforms all tested state-of-the-art approaches.

2 Related Work

2.1 Time Series Anomaly Detection Time series anomaly detection has been extensively studied, with hundreds of methods proposed in recent decades [19]. Due to space constraints, we refer readers to comprehensive surveys [19] for a broader overview and focus here on key approaches that serve as baselines in our work.

Several TSAD methods use Matrix Profile (MP) [25] to detect discords. SCRIMP [26] efficiently computes exact similarity profiles in an anytime fashion, while MERLIN [17] eliminates the need to predefine subsequence lengths. DAMP [15] extends the approach to real-time detection in streaming data, and C^{22}MP [21] integrates catch22 [16] features with MP to enhance feature-based discord detection.

Deep learning methods are popular in recent years. Autoencoder-based models such as LSTM-VAE [18] and USAD [2] learn to reconstruct normal sequences and detect anomalies based on the reconstruction error. Telemanom [9] trains LSTMs to predict future values, and then applies a dynamic threshold to the prediction error to detect anomalies. Transformer-based approaches like TranAD [23] use attention mechanisms to capture long-term dependencies, while adversarial training enhances anomaly detection stability.

Some methods detect anomalies based on density and distribution. RRCF [8] maintains an ensemble of random binary trees, assigning anomaly scores based on data perturbation effects. MDI [3] detects anomalies by identifying subsequences whose distributions differ significantly from the rest of the series using divergence measures. GANF [5] combines normalizing flows with Bayesian networks to model joint probability distributions and detect anomalies as low-density observations. NormA [4] constructs a normal behavior model and detects anomalies based on (dis)similarity.

These methods provide a solid foundation for TSAD and have strong performance, so we incorporate them into our baseline methods.

2.2 Random Convolutional Kernel Random convolutional kernel methods have played an important role in time series classification and clustering in recent years. The first such method, *ROCKET* [6], was proposed by Dempster et al. It applies tens of thousands of convolutional kernels with random weights to input data, calculates two aggregated features, and forms a high-dimensional feature vector. The feature vector is then used for classification using a linear model. MiniRocket [7] streamlines the process by using a small fixed set of kernels with predetermined weights, which reduces variability and significantly improves efficiency while maintaining classification accuracy. MultiRocket [22] further extends MiniRocket by incorporating additional pooling strategies and combining features from multiple kernel sets to capture a richer representation. In time series clustering, Li et al. proposed RandomNet [13], a CNN-LSTM framework with random weights and an ensemble mechanism to filter out irrelevant representations for clustering.

Despite their success in classification and clustering, such a method has not been explored in TSAD. To the best of our knowledge, we are the first to explore using random convolutional kernels for TSAD.

3 Methodology

3.1 Problem Formulation We study semi-supervised time series anomaly detection. Specifically, we consider a univariate time series as the training set, represented as a sequence of observations of size T:

$$\mathcal{T} = \{a_1, \ldots, a_T\},$$

where each data point a_t is collected at a specific timestamp t and $a_t \in \mathbb{R}$. The training dataset consists of historical normal data without anomalies.

Given an unseen test time series $\hat{\mathcal{T}}$ of length $\hat{T}$ with an anomaly, we compute anomaly score sequence $\mathcal{S} = \{s_1, \ldots, s_{\hat{T}}\}$, where each $s_t \in \mathbb{R}^+$ represents the degree

of anomaly at timestamp t. The final anomaly location is determined based on the anomaly score sequence.

3.2 Overview of RandomAD Next, we present a novel anomaly detection framework that leverages random convolutional kernels while incorporating adaptive selection mechanisms for both kernel and window size. As illustrated in Figure 1, our framework preprocesses time series under **multi-window selection strategy** using multiple sliding windows of different sizes. The preprocessed data are then fed into multiple random kernel channels, each corresponding to a specific window size. Each channel consists of three key components: (1) a **random kernel module** that generates various convolutional kernels, (2) an **adaptive kernel selection mechanism** that filters the kernels to extract the most informative features, and (3) a **kNN-based anomaly scoring mechanism** that calculates anomaly scores based on deviations in the feature space. Each channel independently produces an anomaly score sequence, and finally the **anomaly filtering module** selects the final result based on these scores. In the following subsections, we provide a detailed explanation of each component.

3.3 Multi-Window Selection Strategy In time series analysis, selecting an appropriate window size is crucial, as different time series—or even different segments within the same series—can exhibit varying temporal dynamics, which cause some anomalies to only be detectable at specific window sizes [14]. Therefore, relying on a fixed window size, as is common in many existing methods [15, 19, 21, 23], can lead to suboptimal performance. To address this problem, we introduce a multi-window selection strategy that adapts to the intrinsic temporal scales present in the time series data.

Our strategy for window size selection is based on appropriate lower and upper bounds, and then candidate window sizes are generated uniformly within two bounds. The lower bound is set to a fixed value to ensure a minimum context length for capturing subtle local patterns. To determine the upper bound m, we perform an autocorrelation analysis on time series. Specifically, we compute the autocorrelation function for non-negative lags and, within a predefined lag interval (from 10 to 1000), we identify the first significant peak. The peak reflects the dominant periodicity of the series and is selected as the upper bound.

Once the lower and upper bounds are determined, we generate multiple candidate window sizes by evenly dividing the interval. For example, if upper bound $m = 40$ and we choose four candidate window sizes, then the candidate window sizes are 10, 20, 30, and 40 respectively.

Through the integration of multi-window selection strategy, our framework is able to dynamically adapt to the diverse temporal characteristics present in time series data. With the anomaly filtering module, our method provides robust and accurate performance across diverse time series applications. We will discuss the anomaly filtering module in the following subsection.

3.4 Random Kernel-based Feature Extractor Although C^{22}MP [21] achieves good performance, it relies on a handcrafted feature extraction method catch22 [16] to capture time series features, which may ignore some important features inherent in different time series datasets. Inspired by the random convolutional kernel methods [6, 7, 13, 22], we adopt random kernel feature extraction to provide a richer data representation that is capable of capturing a broader range of features. Specifically, we use the random kernel generation mechanism in MiniRocket [7], which we describe in detail below, as it produces shorter feature representations thereby reducing running time.

MiniRocket's kernel generation mechanism uses convolutional kernels of length 9. The weights of each kernel are restricted to two values, -1 and 2, and the sum of the weights equals zero. This zero-sum property ensures that the kernels only focus on relative magnitude in the input rather than absolute values, making them invariant to constant offsets in the data. The bias of each kernel is directly obtained from the convolution output by sampling quantiles from randomly selected training examples. To capture patterns at multiple scales, each kernel is applied with various dilation factors, which spread the kernel across the input sequence. Specifically, a kernel with dilation d processes every d^{th} element of the input. The dilation values are selected from a fixed range $D = \{\lfloor 2^0 \rfloor, \ldots, \lfloor 2^{\max} \rfloor\}$, where the exponents are uniformly spaced between 0 and $\max = \log_2\left(\frac{l_{\text{input}}-1}{l_{\text{kernel}}-1}\right)$, where l_{input} and l_{kernel} are length of input and kernel, respectively. Padding is alternated across kernel/dilation combinations so that half use padding and half do not. Zero padding is added at the start and end of the time series, which ensures the convolution operation begins and ends with the kernel centered on the first and last elements of the sequence. Finally, the extracted features are summarized using Proportion of Positive Values (PPV), which effectively captures the essential characteristics of the convolution output.

3.5 Kernel Selection through Kernel Selection Score The previous module generates a large number

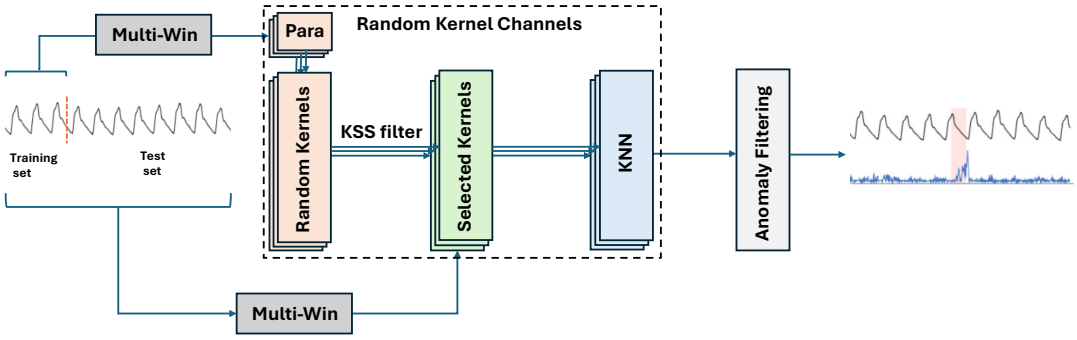

Figure 1: Overview of RandomAD.

of random kernels to extract diverse features from all subsequences. Unlike classification task, where random kernel methods [6, 7, 22] can leverage class labels to learn weights for features through classifier training, semi-supervised anomaly detection relies only on normal data. Without access to labeled anomalies, it requires an effective mechanism to identify informative kernels based on the distribution of normal patterns. To that end, we propose a kernel selection scoring function, which we describe below.

Let $\mathbf{X} \in \mathbb{R}^{N \times M}$ denote the feature matrix, where N represents the number of subsequences extracted via a sliding window, and M is the total number of random kernels. The i-th column, $\mathbf{X}_i$, corresponds to the feature values produced by kernel i across all subsequences.

To address the kernel selection challenge, we introduce Kernel Selection Score (KSS) based on entropy and mutual information. For each kernel i, we first compute its entropy, denoted as:

$$H(\mathbf{X}_i) = -\sum_{x \in \mathcal{V}_i} p_i(x) \log p_i(x), \tag{3.1}$$

where $\mathcal{V}_i$ is the set of feature values from kernel i and $p_i(x)$ is the empirical probability of output x. A lower entropy indicates that the kernel's output is more stable and less noisy.

Next, we quantify the similarity between the outputs of different kernels by computing the mutual information. For any two kernels i and j, the mutual information is defined as:

$$I(\mathbf{X}_i, \mathbf{X}_j) = \sum_{(x,y) \in \mathcal{V}_i \times \mathcal{V}_j} p_{i,j}(x,y) \log \frac{p_{i,j}(x,y)}{p_i(x) p_j(y)}, \tag{3.2}$$

where $p_{i,j}(x,y)$ is the joint probability that kernel i outputs x and kernel j outputs y simultaneously. High mutual information implies that kernel i shares commonality with other kernels, suggesting that it captures underlying patterns within the data.

To integrate these two aspects, we define KSS for kernel i as follows:

$$KSS(\mathbf{X}_i) = \alpha \left(\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} I(\mathbf{X}_i, \mathbf{X}_j) \right) - \beta\, H(\mathbf{X}_i), \tag{3.3}$$

where α and β are positive scaling factors that balance the contributions of mutual information and entropy, respectively. Setting either to 0 removes its corresponding effect. $\mathcal{N}(i)$ denotes the set of all kernels excluding i. The kernels are then ranked in descending order based on their KSS values, and the top γ percent are selected.

By selecting kernels with high KSS values, we can identify random kernels that have both strong mutual information correlation with others and low entropy, thereby being able to extract stable and meaningful patterns from the data. Such method is particularly beneficial in semi-supervised anomaly detection, where the algorithm must rely on the inherent structure of the data.

3.6 kNN-based Anomaly Scoring After selecting the kernel, each channel undergoes feature extraction and anomaly scoring. In each channel, we apply the selected kernel to process the training set and the test set to generate feature vectors to capture the characteristics of each subsequence in the time series.

The anomaly scoring mechanism uses the k-nearest-neighbor (kNN) to quantify the anomaly level of each test subsequence. Specifically, for each test subsequence, we compute its Euclidean distance to all training subsequences in the feature space and identify the k most similar instances. The average Euclidean distance to these top-k neighbors is then assigned as the anomaly score to the last timestamp of the test subsequence.

With random kernels, the distance calculation will focus on the difference of each feature rather than the

Table 1: Comparison of our method against 13 baseline methods on 250 UCR Time Series Anomaly Archive datasets.

Method	Score	Method	Score
RandomAD (Ours)	**0.704**	C^{22}MP	0.568
DAMP	0.556	USAD	0.276
AutoEncoder	0.236	Telemanom	0.468
LSTM-VAE	0.198	SCRIMP	0.416
RRCF	0.030	MERLIN	0.440
MDI	0.470	NormA	0.474
TranAD	0.190	GANF	0.240

difference of raw values in the time series. Finally, each channel will generate an anomaly score sequence.

3.7 Anomaly Filtering To determine the best result from multiple channels, each corresponding to a different candidate window size, we introduce a anomaly filtering mechanism that utilizes the detection index Δ to quantify the difference between anomalous subsequences. For each channel, we first determine the highest anomaly score s_{h1} and the second highest score s_{h2}. Then, we calculate the detection index as

$$\Delta = s_{h1} - s_{h2}. \tag{3.4}$$

A larger Δ indicates a more obvious difference between the most anomalous subsequence and the rest of the data, which indicates that the corresponding window size is more effective in capturing anomalies.

By comparing the detection index of all channels, we select the channel with the largest Δ as the final result. With this mechanism, our framework is able to dynamically adapt to the most appropriate window size, therefore enhancing the robustness and accuracy of the anomaly detection.

4 Experiments

4.1 Experimental Settings In this section, we describe the details of our experimental setup.

Dataset. To evaluate our proposed method, we conduct experiments on various public time series datasets. Given the concerns raised by Wu and Keogh [24] regarding flaws in existing anomaly detection benchmarks, we select datasets that provide realistic and meaningful challenges for anomaly detection. Specifically, we use all 250 datasets from the Hexagon ML/UCR Time Series Anomaly Archive [24], which is introduced to address problem from other benchmark datasets [1, 10, 12, 20] such as trivial anomaly patterns, unrealistic anomaly densities, and mislabeled ground truth. The datasets span diverse domains, including medicine, sports, biology, industry, etc [21, 24]. Each dataset is split into a training set which contains no anomalies, and a test set which contains exactly one labeled anomaly. The sequence lengths and anomaly lengths vary greatly across datasets, with sequence length ranging from 6674 to 900000 data points and anomaly length ranging from 1 to 1701 data points. In addition to this archive, we also use 10 datasets from [21] to intuitively visualize our results.

Evaluation Metrics. To ensure fair and meaningful evaluation, we follow the accuracy metric recommended by the dataset creators and previous works [15, 21, 24]. Specifically, let L be the length of the labeled anomaly, the prediction is considered correct if the location of the highest anomaly score predicted by the algorithm falls within $\pm L$ data points of the ground truth anomaly location. If $L < 100$, we set $L = 100$. The final accuracy score we present is computed as the *ratio of correctly detected anomalies* across all datasets.

Equipment. The experiments are run on a machine with AMD Ryzen 9 5900X and 64 GB RAM. Since the method does not involve neural network training, there is no need to use a GPU.

Hyper-parameters. To balance the contributions of entropy and mutual information, we set both α and β to 1 in Equation 3.5. Each channel uses 1000 random kernels, with a total of 4 channels. We set $\gamma = 0.5$ for the percentage the kernels are selected and for kNN anomaly scoring, we set $k = 3$.

Baseline Methods. we compare our method against 13 state-of-the-art methods: TranAD [23], MDI [3], RRCF [8], LSTM-VAE [18], AutoEncoder [2], USAD [2], Telemanom [9], SCRIMP [26], MERLIN [17], NORMA [4], GANF [5], DAMP [15] and C^{22}MP [21]. See Section 2 for more details.

4.2 Experimental Results Table 1 shows the accuracy scores of 13 baseline methods and our proposed framework on 250 datasets from UCR Anomaly Archive. All results of baseline methods are from [21]. Our framework, RandomAD, achieves an accuracy score of 0.704, outperform all baseline methods and has around a 24% performance improvement over the second-best method, C^{22}MP (0.568).

It is worth noting that in the C^{22}MP paper [21], the authors also presented an ensemble of DAMP and C^{22}MP with a accuracy score of 0.692. However, this approach is not a true algorithm; it is a post-hoc ensemble that selects the better algorithm after manually observing the labels. Our method achieves higher accuracy than this post-hoc ensemble approach without requiring any artificial selection process, and all

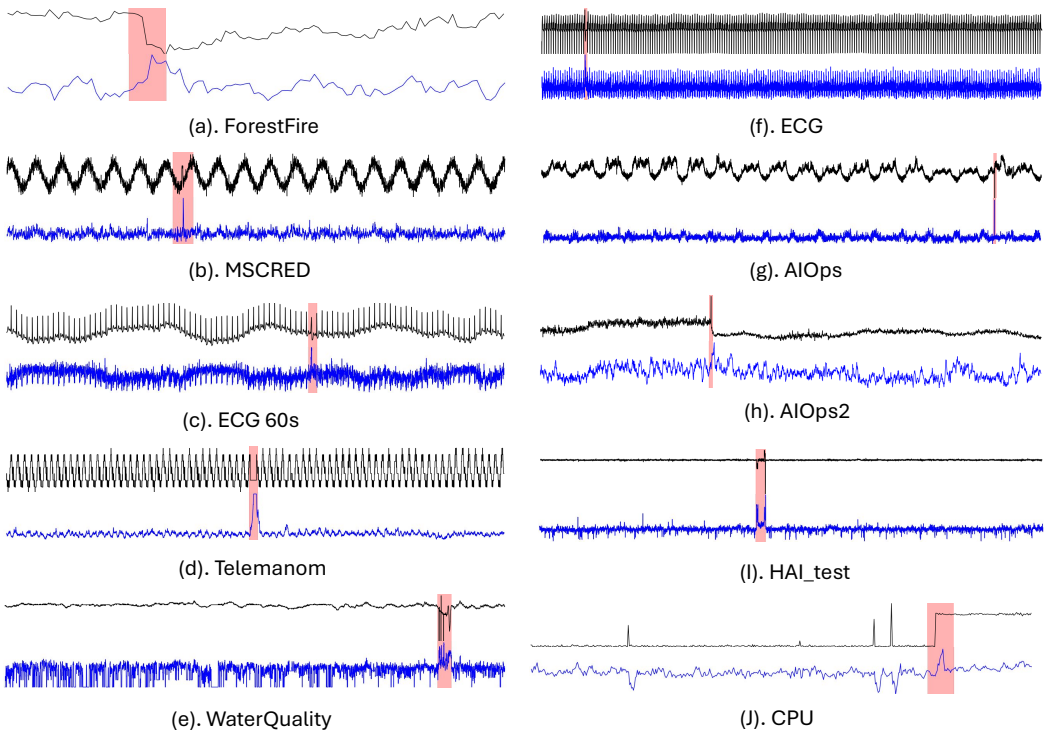

Figure 2: The performance of RandomAD on 10 datasets outside the UCR Anomaly Archive.

modules are automatically adapted.

These results demonstrate the superiority of RandomAD, which leverages random kernel feature extraction, over approaches based on shape comparison and fixed features. The superiority reflects the ability of our random kernel module to effectively capture the underlying patterns in the data, as well as the effectiveness of the multi-window selection and anomaly filtering mechanisms.

4.3 Visualization To demonstrate the effectiveness of our method, Figure 2 presents the visualization of anomalies detected by RandomAD across ten datasets used in previous studies [21], none of which are part of the UCR Anomaly Archive [24]. Each dataset is divided into training set and test set, with the test set containing a single anomaly. The upper portion of each subfigure (shown in black) represents the original test data, while the lower portion (in blue) shows the anomaly scores calculated by our model. The ground-truth anomalies are highlighted with red boxes. Our method predicts the anomaly location based on the highest anomaly score. Our model effectively detects anomalies across all datasets. It is worth noting that even in datasets without clear patterns, such as subfigures (a) and (j), our method can correctly locate the anomalies.

Table 2: Ablation study results show the impact of each module on the performance.

Method Variant	**Score**
RandomAD (Full)	**0.704**
w/ fixed window size (10)	0.548
w/ fixed window size (AT)	0.668
w/o kernel selection	0.688
w/o random kernels	0.580

4.4 Ablation Study To verify the effectiveness of each component in our framework, we conduct an ablation study by removing or replacing individual modules and evaluating the resulting performance. Table 2 summarizes the accuracy scores under various configurations.

As shown in Table 2, the full model achieves the highest score of 0.704. Replacing the multi-window selection strategy with a fixed window size leads to a significant performance drop: with a fixed window size of 10, the score decreases to 0.548, while using a fixed window size based on autocorrelation (AT, equivalent to setting the upper bound as the window size) results in a score of 0.668. This demonstrates the importance of the multi-window selection strategy and the anomaly filtering module in adapting to the varying characteristics of different datasets.

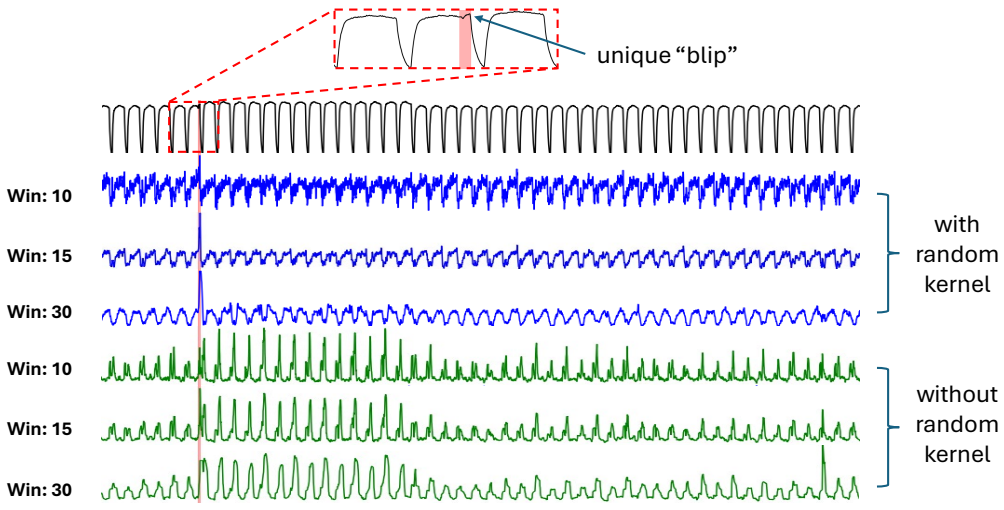

Figure 3: Anomaly detection results with (blue) and without (green) random kernels across different window sizes (10, 15, 30) on the NASA spacecraft dataset. The black line shows a test segment with an anomaly (”blip” in red box).

In addition, removing the kernel selection mechanism leads to a slight drop in performance (to 0.688). It is worth noting that since removing this mechanism significantly increases the feature length, it also significantly increases the total running time (from 3.7 hours to 7.3 hours). The result highlights the effectiveness of this mechanism in selecting more informative kernels.

Finally, removing random kernels and applying kNN directly on the raw subsequences for anomaly detection reduces the accurcay score to 0.580, illustrating the effectiveness of random convolutional kernels in capturing meaningful features.

4.5 Effectiveness Analysis To further illustrate the advantages of using random kernels and demonstrate that our method is less sensitive to small changes in window size, we conduct experiments on a real dataset from NASA spacecraft. The top part of Figure 3 (black line) represents a segment of the test set, where we highlight the anomaly, a small unique ”blip” of length 15, within the red dashed box. The blue line in the figure shows the anomaly scores produced by our method under three different window sizes: 10, 15, and 30. The green line represents the results when removing the random kernel feature extraction and relying on raw subsequences for kNN-based anomaly scoring.

The results clearly demonstrate that with random kernels, all three window sizes successfully detect the anomaly. In contrast, when using raw values, only the window size of 15 correctly detects the anomaly, while the others fail. Moreover, without random kernels, the anomaly score struggles to differentiate abnormal subsequences from normal ones. For instance, with a window size of 30, some normal subsequences on the right side exhibit high anomaly scores. In comparison, the random kernel method maintains clear separation between normal and abnormal subsequences across all window sizes and the normal regions do not experience abrupt changes in anomaly scores due to changes in window size.

The comparison highlights two advantages of our method. First, the random kernels selected by the kernel selection mechanism can effectively extract meaningful features that can capture the underlying patterns. Second, this method is less sensitive to the window size, allowing us to search the appropriate window size in a small search space.

5 Conclusion

In this work, weintroduce RandomAD, the first random convolutional kernel method for time series anomaly detection. While the idea of using convolutional kernels with random weights to generate diverse features is not new, the main contribution of our work lies in the kernel selection mechanism to select informative kernels that capture meaningful pattern in the sequence. In addition, since the random kernel module is robust to small variations in window size, we introduce an efficient multi-window selection strategy with a small search space. Incorporated with the anomaly filtering mechanism, our method effectively determines the appropriate window size and final anomalies. Extensive experiments on 250 datasets from the UCR Anomaly Archive demonstrate that RandomAD outperforms state-of-the-art methods.

References

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

[2] J. Audibert, S. Marti, F. Guyard, and M. A. Zuluaga. From univariate to multivariate time series anomaly detection with non-local information. In *Advanced Analytics and Learning on Temporal Data: 6th ECML PKDD Workshop, AALTD 2021, Bilbao, Spain, September 13, 2021, Revised Selected Papers 6*, pages 186–194. Springer, 2021.

[3] B. Barz, E. Rodner, Y. G. Garcia, and J. Denzler. Detecting regions of maximal divergence for spatio-temporal anomaly detection. *IEEE transactions on pattern analysis and machine intelligence*, 41(5):1088–1101, 2018.

[4] P. Boniol, M. Linardi, F. Roncallo, T. Palpanas, M. Meftah, and E. Remy. Unsupervised and scalable subsequence anomaly detection in large data series. *The VLDB Journal*, 30(6):909–931, 2021.

[5] E. Dai and J. Chen. Graph-augmented normalizing flows for anomaly detection of multiple time series. *arXiv preprint arXiv:2202.07857*, 2022.

[6] A. Dempster, F. Petitjean, and G. I. Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020.

[7] A. Dempster, D. F. Schmidt, and G. I. Webb. Minirocket: A very fast (almost) deterministic transform for time series classification. In *Proceedings of the 27th ACM SIGKDD*, pages 248–257, 2021.

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

[9] 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*, pages 387–395, 2018.

[10] 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*, pages 387–395, 2018.

[11] 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.

[12] N. Laptev, S. Amizadeh, and Y. Billawala. S5-a labeled anomaly detection dataset, version 1.0 (16m), 2015.

[13] X. Li, W. Xi, and J. Lin. Randomnet: clustering time series using untrained deep neural networks. *Data Mining and Knowledge Discovery*, 38(6):3473–3502, 2024.

[14] Y. Lu, T. V. A. Srinivas, T. Nakamura, M. Imamura, and E. Keogh. Matrix profile xxx: Madrid: A hyper-anytime and parameter-free algorithm to find time series anomalies of all lengths. In *ICDM 2023*, pages 1199–1204. IEEE, 2023.

[15] Y. Lu, R. Wu, A. Mueen, M. A. Zuluaga, and E. Keogh. Damp: accurate time series anomaly detection on trillions of datapoints and ultra-fast arriving data streams. *Data Mining and Knowledge Discovery*, 37(2):627–669, 2023.

[16] C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher, and N. S. Jones. catch22: Canonical time-series characteristics: Selected through highly comparative time-series analysis. *Data Mining and Knowledge Discovery*, 33(6):1821–1852, 2019.

[17] T. Nakamura, M. Imamura, R. Mercer, and E. Keogh. Merlin: Parameter-free discovery of arbitrary length anomalies in massive time series archives. In *ICDM 2020*, pages 1190–1195. IEEE, 2020.

[18] D. Park, Y. Hoshi, and C. C. Kemp. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters*, 3(3):1544–1551, 2018.

[19] S. Schmidl, P. Wenig, and T. Papenbrock. Anomaly detection in time series: a comprehensive evaluation. *Proceedings of the VLDB Endowment*, 15(9):1779–1797, 2022.

[20] 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*, pages 2828–2837, 2019.

[21] S. Tafazoli, Y. Lu, R. Wu, T. V. A. Srinivas, H. D. Cruz, R. Mercer, and E. Keogh. Matrix profile xxix: C 22 mp, fusing catch 22 and the matrix profile to produce an efficient and interpretable anomaly detector. In *ICDM 2023*, pages 568–577. IEEE, 2023.

[22] C. W. Tan, A. Dempster, C. Bergmeir, and G. I. Webb. Multirocket: multiple pooling operators and transformations for fast and effective time series classification. *Data Mining and Knowledge Discovery*, 36(5):1623–1646, 2022.

[23] S. Tuli, G. Casale, and N. R. Jennings. Tranad: deep transformer networks for anomaly detection in multivariate time series data. *Proceedings of the VLDB Endowment*, 15(6):1201–1214, 2022.

[24] R. Wu and E. J. Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE transactions on knowledge and data engineering*, 35(3):2421–2429, 2021.

[25] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh. Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In *ICDM 2016*, pages 1317–1322. Ieee, 2016.

[26] 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 *ICDM 2018*, pages 837–846. IEEE, 2018.