RandomAD: A Random Kernel-Based Anomaly Detector for Time Series

Wenjie Xi(✉) and Jessica Lin

George Mason University, Fairfax, VA 22033, USA
{wxi,jessica}@gmu.edu

Abstract. Time series anomaly detection is a critical task with a wide range of applications including industrial monitoring, financial fraud detection, and medical diagnostics. Among existing methods, C^{22}MP represents the state-of-the-art by combining Matrix Profile with catch22, a hand-crafted feature set, to enhance anomaly detection performance. However, catch22 features are limited in their ability to capture a full range of temporal characteristics in time series data. Recent advances in random convolutional kernel methods, such as the ROCKET family, have demonstrated strong performance in time series classification and clustering tasks. In this work, we propose RandomAD, a semi-supervised anomaly detection approach that leverages thousands of random convolutional kernels to extract a rich set of features. Our method adopts MiniRocket's random kernel generation strategy to produce a large pool of kernels with randomly initialized weights based on the training data. To address the lack of labeled anomalies in the semi-supervised setting, we introduce a kernel selection mechanism to retain only the most informative kernels. Additionally, we incorporate a multi-window selection strategy with an anomaly filtering module to optimize both window size and detection results. Through extensive experiments on the benchmark datasets, we demonstrate that RandomAD consistently outperforms existing state-of-the-art methods.

Keywords: Random convolutional kernel · Anomaly detection · Time series

1 Introduction

Time series anomaly detection (TSAD) focuses on detecting unusual subsequences (often referred to as *discords* when they represent the most anomalous patterns) or time points within long data streams. Due to its significance in wide-ranging applications including industrial monitoring, fraud detection and medical diagnostics [10], it has attracted considerable attention from researchers [19]. One remarkable approach for finding discords is the Matrix Profile (MP) [27], a data structure that stores each subsequence's Euclidean distance to its nearest neighbor. MP-based methods [14,27] are effective at finding anomalies that stand out due to their distinctive shapes.

R. P. Ribeiro et al. (Eds.): ECML PKDD 2025, LNAI 16013, pp. 159–175, 2026.
https://doi.org/10.1007/978-3-032-05962-8_10

However, relying solely on shape-based comparisons has limitations. In many real-world applications, anomalies may not manifest as distinct shapes but instead as subtle deviations in statistical characteristics or underlying dynamics. Incorporating features 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 [15], 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) it requires domain knowledge or labeled data when selecting the most relevant features for a specific dataset.

Inspired by recent success of random convolutional kernel techniques in time series classification [6,7,22] and clustering [12], we propose RandomAD, which extracts a rich set of features using 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. Unlike MiniRocket, which is a supervised method that leverages class labels to guide kernel selection, the semi-supervised anomaly detection does not have access to labeled anomalies. Therefore, we propose a kernel selection mechanism to retain only the most informative kernels capable of effectively capturing underlying patterns under semi-supervised setting. 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 lengths or manually tuned "magic numbers", which are often suboptimal when handling different types of anomalies with varying characteristics [13]. Several approaches have been proposed to address this problem. For example, MADRID [13] 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 subsequences 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 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 Sect. 4.6. 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:

- We introduce random convolutional 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 an anomaly filtering mechanism that automatically determines the optimal window size and final anomaly detection result based on the 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 constraint, we refer readers to the comprehensive survey [19] for a broader overview and focus here on key approaches that serve as baselines in our work.

Matrix Profile (MP) [27], a data structure that stores the Euclidean distance of each subsequence to its nearest neighbor, is a widely used method for time series anomaly detection. Intuitively, the matrix profile can be used to detect discord—the most unusual subsequence in the time series—by identifying the subsequence with the largest nearest neighbor distance. Since its initial introduction, many variants and follow-up works have been proposed to enhance its capabilities. SCRIMP [28] efficiently computes exact MPs in an anytime fashion, while MERLIN [17] eliminates the need to predefine subsequence lengths. DAMP [14] extends MP to real-time detection in streaming data, and C^{22}MP [21] integrates catch22 [15] features with MP to detect anomalies by using time series characteristics instead of shapes.

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 [24] 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] builds a model of normal behavior and detects anomalies by measuring (dis)similarity to this reference model.

These methods offer a strong foundation for time series anomaly detection and demonstrate competitive performance, making them well-suited for inclusion as baseline comparisons in our evaluation.

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 [12], a CNN-LSTM framework with random weights and an ensemble mechanism to filter out irrelevant representations for clustering.

ROCKET has also been adopted for anomaly detection with classical detectors such as One-Class SVM [16], Isolation Forest [29], and kNN variants [23]. We extend this direction by using MiniRocket and introducing kernel selection and multi-window selection with an anomaly filtering mechanism for semi-supervised TSAD.

3 Methodology

3.1 Problem Formulation

We investigate the problem of 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:

$$\mathcal{T} = \{a_1, \ldots, a_T\}, \tag{1}$$

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.

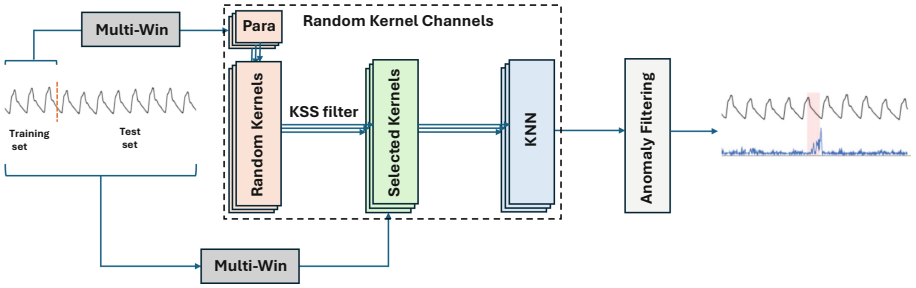


Fig. 1. Overview of RandomAD.

3.2 Overview of RandomAD

Next, we present a novel anomaly detection framework that leverages random convolutional kernels with adaptive selection mechanisms to select the best kernels and window sizes. As illustrated in Fig. 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, making some anomalies detectable only at specific window sizes [13]. Therefore, relying on a fixed window size, as is common in many existing methods [14,19,21,24], 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 window size selection strategy defines appropriate lower and upper bounds, within which candidate window sizes are generated uniformly. 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 the upper bound $m = 40$ and we select four candidate window sizes, the resulting candidate window sizes would be 10, 20, 30, and 40.

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] demonstrates strong performance, it relies on a handcrafted feature extraction method catch22 [15] to capture time series characteristics, which may miss important features unique to different time series datasets. Inspired by the random convolutional kernel methods [6,7,12,22], 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 has been shown to produce a smaller yet effective feature representations in a fraction of time.

Length and Weights. 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.

Bias. The bias of each kernel is directly obtained from the convolution output by sampling quantiles from randomly selected training examples.

Dilation. 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. 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.

Feature Extraction. 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 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:

$$H(\mathbf{X}_i) \;=\; -\sum_{x \,\in\, \mathcal{V}_i} p_i(x)\, \log p_i(x), \tag{2}$$

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:

$$I(\mathbf{X}_i, \mathbf{X}_j) = \sum_{(x,y) \in \mathcal{V}_i \times \mathcal{V}_j} p_{i,j}(x,y) \log \frac{p_{i,j}(x,y)}{p_i(x) p_j(y)}, \tag{3}$$

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 another kernel, suggesting that it captures underlying patterns within the data.

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

$$KSS(\mathbf{X}_i) = \alpha \left(\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} I(\mathbf{X}_i, \mathbf{X}_j) \right) - \beta\, H(\mathbf{X}_i), \tag{4}$$

where $\mathcal{N}(i)$ denotes the set of all kernels excluding i. α and β are positive scaling factors that balance the contributions of mutual information and entropy, respectively. Entropy and mutual information are closely related: entropy measures the stability of features generated by a single kernel, while mutual information evaluates the relationships between different kernels. Integrating both metrics enables the selection of kernels that are both stable and share meaningful commonalities. Setting either parameter to 0 eliminates its corresponding effect. We evaluate the

impact of each in the following section. The kernels are then ranked in descending order based on their KSS values, and the top γ fraction are selected.

By selecting kernels with high KSS values, we can identify random kernels that have both strong mutual information with others and low entropy, thereby extracting stable and meaningful patterns from the data. Such a 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. For each channel, we apply the selected kernel to both the training and test sets, and generate feature vectors to capture the characteristics of each subsequence in the time series.

The anomaly scoring mechanism employs the k-nearest-neighbor (kNN) algorithm 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 nearest neighbors. The anomaly score assigned to the last timestamp of the test subsequence is the average Euclidean distance to these top-k neighbors.

Using random kernels, the distance calculation focuses on the difference of the features rather than raw time series values. As a result, each channel produces its own sequence of anomaly scores.

3.7 Anomaly Filtering

To determine the best result from multiple channels, each corresponding to a different candidate window size, we introduce an anomaly filtering mechanism that utilizes the detection index Δ to quantify the difference between anomalous subsequences. For each channel, we obtain a sequence of anomaly scores, $\mathcal{S} = \{s_1, s_2, \ldots, s_{\hat{T}}\}$, computed with a candidate window size w. The filtering process is performed as follows:

1. **Identifying the Primary Anomaly:**
   We first locate the highest anomaly score s_{h1} in $\mathcal{S}$ and its corresponding index i_{h1}.
2. **Defining an Exclusion Range:**
   To avoid selecting nearby points that may belong to the same anomaly, we define an exclusion range centered around i_{h1}. Specifically, we set:

$$EX_{start} = \max\{0,\, i_{h1} - w\}, \tag{5}$$

$$EX_{end} = \min\{\hat{T},\, i_{h1} + w\}. \tag{6}$$

   Within the interval $[EX_{start},\, EX_{end}]$, none of the scores can be further considered.

3. **Computing the Detection Index:**
   Next, we search for the highest anomaly score outside the exclusion range, denoted as s_{h2}. The detection index is then calculated by:

$$\Delta = s_{h1} - s_{h2}. \tag{7}$$

   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.
4. **Channel Selection:**
   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 [26] 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 [25], which was introduced to address the problems from other benchmark datasets [1,9,11,20] such as trivial anomaly patterns, unrealistic anomaly densities, and mislabeled ground truth. The datasets span diverse domains, including medicine, sports, biology, industry, etc. [21,26]. 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 [14, 21,26]. 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.

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

Method	Score	Method	Score
RandomAD (Full)	**0.704**	DAMP	0.556
RandomAD ($\alpha = 0$)	0.688	C^{22}MP	0.568
RandomAD ($\beta = 0$)	0.688	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

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

Baseline Methods. We compare our method against 13 state-of-the-art methods: TranAD [24], MDI [3], RRCF [8], LSTM-VAE [18], AutoEncoder [2], USAD [2], Telemanom [9], SCRIMP [28], MERLIN [17], NormA [4], GANF [5], DAMP [14] and C^{22}MP [21]. See Sect. 2 for more details. Our source code is publicly available[1].

4.2 Experimental Results

Table 1 shows the accuracy scores of our proposed approach and 13 baseline methods on 250 datasets from the UCR Anomaly Archive. All results of baseline methods are from [21]. Due to space constraint, we are unable to show the selected window size and anomaly detection result of our method on each dataset. Interested readers can refer to our GitHub repository[1] for the full results.

From Table 1, we can observe that our proposed work, RandomAD, achieves an accuracy score of 0.704, which outperforms all baseline methods and has around a 24% performance improvement over the second-best method, C^{22}MP (0.568). Additionally, we include results where the impact of mutual information (RandomAD, $\alpha = 0$) and entropy (RandomAD, $\beta = 0$) are individually removed in Eq. 4. Both have anomaly score of 0.688 which shows the importance of combining both in kernel selection.

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 an 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 without relying on any manual selection processes, as all modules are automatically adapted to the data.

These results demonstrate the superiority of RandomAD, which leverages random kernel feature extraction, over approaches based on shape comparison

[1] https://github.com/Jackxiini/RandomAD

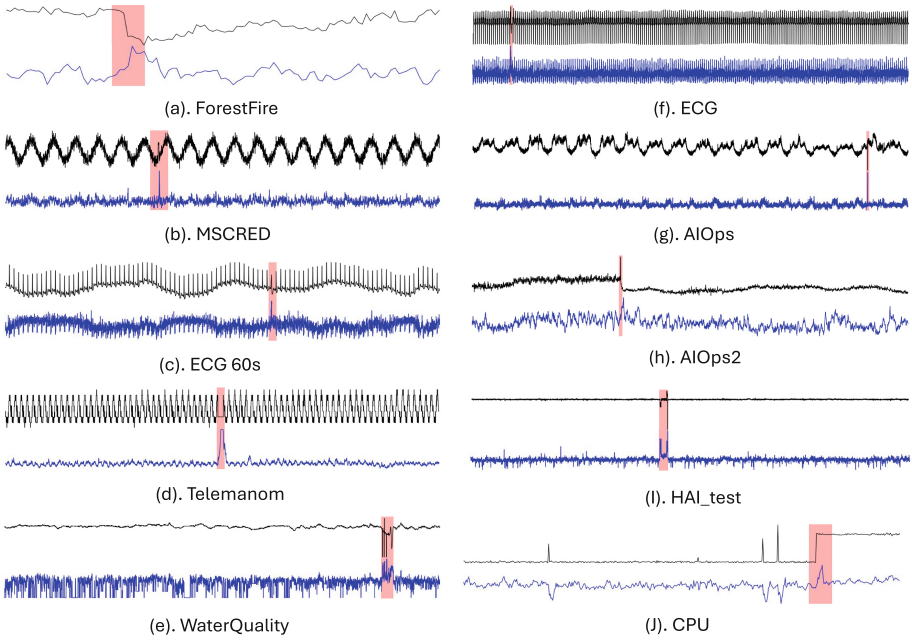


Fig. 2. The performance of RandomAD on 10 datasets outside the UCR Anomaly Archive. The top portion (black) is the original test data, and the bottom portion (blue) is the corresponding anomaly score computed by RandomAD. Ground-truth anomalies are marked by red boxes. (Color figure online)

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, Fig. 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 [25]. 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 effectively detects anomalies and their correct locations in 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.

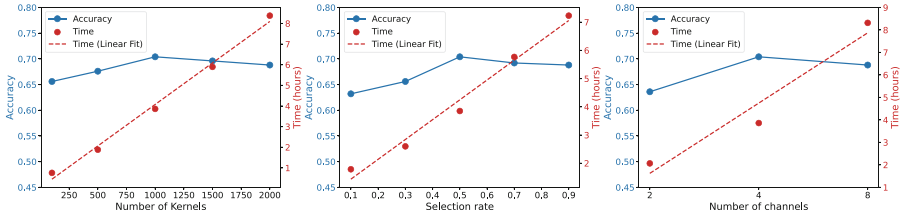
In addition, removing the kernel selection mechanism leads to a slight drop in performance (to 0.688). It is worth noting that since removing this mechanism significantly increases the feature length, it significantly increases the total running time (from 3.8 h to 7.3 h). The result highlights the effectiveness 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 accuracy score to 0.580, illustrating the effectiveness of random convolutional kernels in capturing meaningful features.

4.5 Sensitivity Analysis

To investigate the sensitivity of our method's performance to hyperparameters, we evaluated five hyperparameters: number of kernels, kernel selection rate γ, number of channels and the scaling factors in Eq. 4, α and β. Fig. 3 shows the results for the first three hyperparameters, along with a linear regression curve fit demonstrating how the running time differs from the linear trend.

Number of Kernels. The number of generated kernels directly affects the ability of our method to extract features. Figure 3a demonstrates that increasing the number of kernels generally improves accuracy; however, this improvement



(a) Effect of the number of kernels on accuracy and running time.

(b) Effect of the kernel selection rate on accuracy and running time.

(c) Effect of the number of channels on accuracy and running time.

Fig. 3. Sensitivity analysis illustrating the trade-offs between accuracy and computational cost under different configurations.

stops after reaching a certain value. Specifically, the accuracy score increases from 0.652 at 100 kernels to a peak of 0.704 at 1000 kernels, and then decreases slightly to 0.688 at 2000 kernels. This suggests that exceeding a certain number of kernels may generate too many irrelevant kernels, which can hurt performance.

In contrast, the running time increases almost linearly with the number of kernels, from 0.76 h with 100 kernels to 8.38 h with 2000 kernels. While using more kernels (e.g.,1000 kernels) can improve performance, choosing fewer kernels can significantly reduce the computational cost. Notably, choosing only 100 kernels reduces the running time by about one-tenth compared to 1000 kernels, while still maintaining competitive accuracy. It is also worth mentioning that even with only 100 kernels, the accuracy score of 0.652 is still significantly higher than other compared methods. This demonstrates the superiority of using random convolutional kernels.

Kernel Selection Rate. Choosing an appropriate kernel selection rate is critical to performance. A selection rate that is too low may discard many informative kernels, while a selection rate that is too high may introduce irrelevant kernels, both of which have a negative impact on accuracy. Figure 3b shows that as the kernel selection rate increases, the accuracy improves until about 0.5, after which the accuracy decreases slightly. In addition, the computation time increases nearly linearly with the increase in the selection rate. Therefore, we suggest using 0.5 as the kernel selection rate to balance accuracy and computational efficiency.

Number of Channels. Increasing the number of window size candidates in the Multi-Window Selection Strategy directly corresponds to an increased number of channels, with each window size corresponding to a separate channel. Figure 3c demonstrates the impact of varying the number of channels on performance. The accuracy score improves significantly when the number of channels increases from 2 to 4, but additional channels (from 4 to 8) do not continue to improve

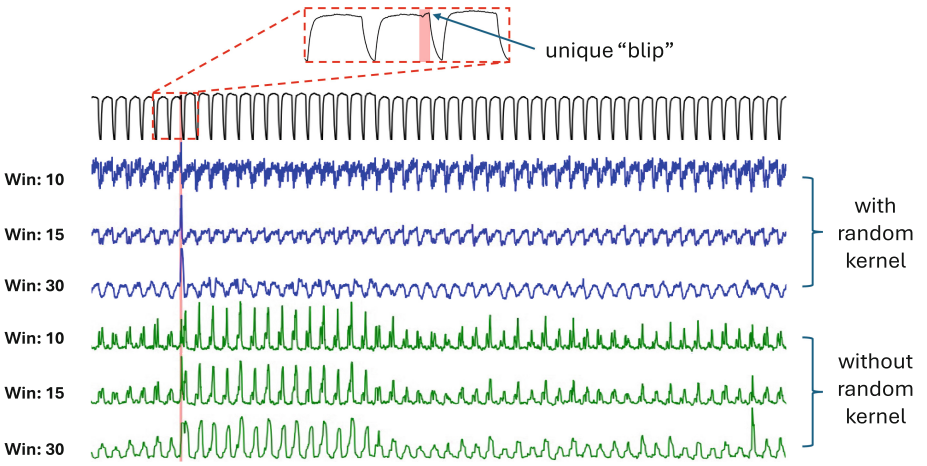


Fig. 4. Anomaly detection results with (blue) and without (green) random kernels across different window sizes (10, 15, 30) on the NASA spacecraft dataset. The black line shows a test segment with an anomaly ("blip" in red box). (Color figure online)

the accuracy score. This demonstrates the effectiveness of our proposed multi-window selection strategy and anomaly filtering mechanism, and also shows that our method is insensitive to small changes in the window size, which allows our method to provide good performance with a small set of window size candidates. We further analyze this in the following section.

Scaling Factor. We further analyze the sensitivity of the accuracy score to α and β. We have highlighted the importance of balancing these two factors, as demonstrated in Table 1, where setting either α or β to 0 leads to performance degradation. To further investigate their influence, we conduct additional experiments by setting $\alpha = 0.2$ and $\beta = 0.8$, which obtain an accuracy score of 0.672. Reversing these values ($\alpha = 0.8$, $\beta = 0.2$) achieves a similar accuracy score of 0.66. These results indicate that both factors significantly affect performance. Given that assigning equal values produces the best result, we recommend setting α and β equally to balance their contributions.

4.6 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 Fig. 4 (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, we introduce RandomAD, a 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. Ahmad, S., Lavin, A., Purdy, S., Agha, Z.: Unsupervised real-time anomaly detection for streaming data. Neurocomputing **262**, 134–147 (2017)
2. Audibert, J., Marti, S., Guyard, F., Zuluaga, M.A.: From univariate to multivariate time series anomaly detection with non-local information. In: Lemaire, V., Malinowski, S., Bagnall, A., Guyet, T., Tavenard, R., Ifrim, G. (eds.) AALTD 2021. LNCS (LNAI), vol. 13114, pp. 186–194. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-91445-5_12
3. Barz, B., Rodner, E., Garcia, Y.G., Denzler, J.: Detecting regions of maximal divergence for spatio-temporal anomaly detection. IEEE Trans. Pattern Anal. Mach. Intell. **41**(5), 1088–1101 (2018)
4. Bhatnagar, A., et al.: Merlion: a machine learning library for time series. arXiv preprint arXiv:2109.09265 (2021)

5. Dai, E., Chen, J.: Graph-augmented normalizing flows for anomaly detection of multiple time series. arXiv preprint arXiv:2202.07857 (2022)
6. Dempster, A., Petitjean, F., Webb, G.I.: Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. Data Min. Knowl. Disc. **34**(5), 1454–1495 (2020)
7. Dempster, A., Schmidt, D.F., Webb, G.I.: MINIROCKET: a very fast (almost) deterministic transform for time series classification. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 248–257 (2021)
8. Guha, S., Mishra, N., Roy, G., Schrijvers, O.: Robust random cut forest based anomaly detection on streams. In: International Conference on Machine Learning, pp. 2712–2721. PMLR (2016)
9. Hundman, K., Constantinou, V., Laporte, C., Colwell, I., Soderstrom, T.: 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)
10. Jacob, V., Song, F., Stiegler, A., Rad, B., Diao, Y., Tatbul, N.: Exathlon: a benchmark for explainable anomaly detection over time series. arXiv preprint arXiv:2010.05073 (2020)
11. Laptev, N., Amizadeh, S., Billawala, Y.: S5-a labeled anomaly detection dataset, version 1.0 (16m) (2015)
12. Li, X., Xi, W., Lin, J.: RandomNet: clustering time series using untrained deep neural networks. Data Min. Knowl. Disc. **38**(6), 3473–3502 (2024)
13. Lu, Y., Srinivas, T.V.A., Nakamura, T., Imamura, M., Keogh, E.: Matrix profile XXX: MADRID: a hyper-anytime and parameter-free algorithm to find time series anomalies of all lengths. In: 2023 IEEE International Conference on Data Mining (ICDM), pp. 1199–1204. IEEE (2023)
14. Lu, Y., Wu, R., Mueen, A., Zuluaga, M.A., Keogh, E.: DAMP: accurate time series anomaly detection on trillions of datapoints and ultra-fast arriving data streams. Data Min. Knowl. Disc. **37**(2), 627–669 (2023)
15. Lubba, C.H., Sethi, S.S., Knaute, P., Schultz, S.R., Fulcher, B.D., Jones, N.S.: Catch22: canonical time-series characteristics: selected through highly comparative time-series analysis. Data Min. Knowl. Disc. **33**(6), 1821–1852 (2019)
16. Melakhsou, A.A., Batton-Hubert, M.: Explainable abnormal time series subsequence detection using random convolutional kernels. In: International Conference on Deep Learning Theory and Applications, pp. 280–294. Springer (2023)
17. Nakamura, T., Imamura, M., Mercer, R., Keogh, E.: MERLIN: parameter-free discovery of arbitrary length anomalies in massive time series archives. In: 2020 IEEE International Conference on Data Mining (ICDM), pp. 1190–1195. IEEE (2020)
18. Park, D., Hoshi, Y., Kemp, C.C.: A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. IEEE Robot. Autom. Lett. **3**(3), 1544–1551 (2018)
19. Schmidl, S., Wenig, P., Papenbrock, T.: Anomaly detection in time series: a comprehensive evaluation. Proc. VLDB Endow. **15**(9), 1779–1797 (2022)
20. Shu, X., Zhang, S., Li, Y., Chen, M.: An anomaly detection method based on random convolutional kernel and isolation forest for equipment state monitoring. Eksploatacja i Niezawodność **24**(4), 758–770 (2022)
21. Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., Pei, D.: Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In: Proceed-

ings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2828–2837 (2019)
22. Tafazoli, S., et al.: Matrix profile XXIX: C 22 MP, fusing catch 22 and the matrix profile to produce an efficient and interpretable anomaly detector. In: 2023 IEEE International Conference on Data Mining (ICDM), pp. 568–577. IEEE (2023)
23. Tan, C.W., Dempster, A., Bergmeir, C., Webb, G.I.: MultiRocket: multiple pooling operators and transformations for fast and effective time series classification. Data Min. Knowl. Disc. **36**(5), 1623–1646 (2022)
24. Theissler, A., Wengert, M., Gerschner, F.: Rockad: transferring rocket to whole time series anomaly detection. In: International Symposium on Intelligent Data Analysis, pp. 419–432. Springer (2023)
25. Tuli, S., Casale, G., Jennings, N.R.: TranAD: deep transformer networks for anomaly detection in multivariate time series data. Proc. VLDB Endow. **15**(6), 1201–1214 (2022)
26. Wu, R., Keogh, E.J.: Supporting page for current time series anomaly detection benchmarks are flawed and are creating the illusion of progress (2020). https://wu.renjie.im/research/anomaly-benchmarks-are-flawed/tkde/
27. Wu, R., Keogh, E.J.: Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. IEEE Trans. Knowl. Data Eng. **35**(3), 2421–2429 (2021)
28. Yeh, C.C.M., et al.: Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 1317–1322. IEEE (2016)
29. Zhu, Y., Yeh, C.C.M., Zimmerman, Z., Kamgar, K., Keogh, E.: Matrix profile XI: SCRIMP++: time series motif discovery at interactive speeds. In: 2018 IEEE International Conference on Data Mining (ICDM), pp. 837–846. IEEE (2018)