Dynamic Anomaly Detection Using Robust Random Cut Forests in Resource-Constrained IoT Environments

Sristi Vashisth, Anjali Goyal
Sharda University, Department of Computer Science and Engineering, Greater Noida, India
E-mail: srishtivashisht1509@gmail.com

Keywords: anomaly detection, robust random cut forest, dynamic data streams, resource constrained IoT environment

Received: August 5, 2024

This paper investigates dynamic anomaly detection in resource-constrained environments by leveraging Robust Random Cut Forests (RRCF). Anomaly detection is crucial for maintaining the integrity and security of data streams in Internet of Things (IoT) environments, where data is continuously generated and often subject to noise and fluctuations. We begin with a comprehensive exploration of resilient random cut data structures tailored for analyzing incoming data streams, highlighting their effectiveness in adapting to the dynamic nature of IoT.Our methodology encompasses extensive experimentation with diverse datasets, including real-time Arduino data and benchmark datasets such as IoT-23 and CIC-IoT. Through this approach, we assess the performance of the RRCF algorithm under various scenarios, focusing on its capability to accurately identify trends and anomalies over time. Notably, we achieve significant performance improvements, with an average Area Under the Curve (AUC) of 95.6 and an F1 score of 0.86, demonstrating RRCF's effectiveness in real-time anomaly detection.To further enhance detection accuracy, we introduce dynamic thresholds that adapt to changing data characteristics, allowing our model to maintain robust performance even in the presence of noise. Detailed evaluations reveal that our approach consistently outperforms existing state-of-the-art methods, particularly in terms of handling noisy data and ensuring computational efficiency under resource constraints.The findings underscore the potential of RRCF as a powerful tool for real-time applications within IoT systems, providing a solid theoretical foundation for future advancements in dynamic anomaly detection. By investigating non-parametric anomalies and analyzing the influence of external factors on data integrity, we uncover hidden patterns amidst dynamic fluctuations. This research emphasizes the need for adaptive strategies in evolving data landscapes, laying the groundwork for enhanced resilience and accuracy in anomaly detection methodologies. In summary, this study presents a novel approach that integrates theoretical insights, updating strategies, and empirical experimentation, making a valuable contribution to the field of anomaly detection in resource-constrained environments. The implications of our work extend beyond theoretical foundations, offering practical solutions for real-time monitoring and anomaly detection in complex, dynamic systems.

Povzetek: V prispevku je predstavljen algoritem Robust Random Cut Forest (RRCF) za dinamično zaznavanje anomalij v IoT okoljih z omejenimi viri, ki izboljša natančnost in učinkovitost detekcije.

1 Introduction

Recent advancements in the Internet of Things (IoT) have transformed urban management and environmental monitoring. For example, Bourougaa-Tria et al. (2021)[13] developed the SPubBin, a smart waste management solution that employs deep learning for efficient waste sorting, addressing urban waste challenges. Similarly, Ali (2021)[14] proposed an IoT-based framework for real-time air quality monitoring in smart cities. These innovations highlight the need for adaptive technologies, such as Robust Random Cut Forests, to effectively detect anomalies and support sustainable urban environments. Anomaly detection presents a significant challenge in data mining, particularly with the proliferation of data generated by sensors and the Internet of Things (IoT). Unlike traditional datasets, data streams in IoT environments are inherently dynamic, complicating the identification of consistent patterns. This raises two fundamental questions: how do we identify anomalies, and what methodologies are most effective for analyzing such data?

To identify outliers, we must assess the complexity of the data. An outlier is defined as an observation that contributes disproportionately to the complexity of the dataset. We must also consider the phenomenon of "outlier masking," whereby the presence of duplicate observations obscures the detection of true outliers. Rapid identification methods are critical for managing continuously evolving data streams.

Additionally, we evaluate a methodology proposed by Liu et al. [2], which has been noted for its effectiveness by Emmott et al. [3]. Although this approach is relatively novel in the context of anomaly detection, randomization techniques are already well-established in various forms of

machine learning. However, Liu et al.'s method exhibits limitations, particularly in its ability to detect significant anomalies amidst high dimensionality. Consequently, alternative strategies, such as those advanced by Tan et al. [4], have also demonstrated shortcomings in effectively handling streaming data.

The inherently variable nature of data streams complicates anomaly detection efforts, underscoring the necessity of developing robust methodologies that can adapt to these changes. Traditional analytical approaches often fail to perform adequately in this context. While Liu et al.'s method shows potential, its limitations—especially regarding high dimensionality—must be addressed. Previous approaches, such as those proposed by Tan et al., have similarly struggled with identifying anomalies in dynamic datasets.

Real-time anomaly detection necessitates efficient mechanisms for assessing the complexity of data as it evolves. This implies that any effective methodology must be capable of accommodating updates and changes with agility. The success of an anomaly detection system is contingent upon its ability to keep pace with incoming data. Exploration of supplementary domains, such as data simplification and unsupervised pattern recognition, may yield advancements in anomaly detection methodologies.

Furthermore, it is important to explore the applicability of these techniques across various sectors, including finance, healthcare, and cybersecurity. Anomaly detection frameworks must be tailored to meet the specific requirements of each industry.

Therefore, the detection of anomalies in dynamic datasets is a formidable challenge that necessitates the development of innovative approaches. We require methodologies that can swiftly adapt to evolving data while maintaining effectiveness in anomaly detection. Although randomized techniques appear promising, further research is essential to enhance their applicability to streaming data. Drawing insights from other fields and addressing real-world challenges will contribute to the advancement of anomaly detection systems in the future.

2 Background and related work

An essential aspect of many applications is anomaly detection, which requires sophisticated algorithms capable of handling the complexities inherent in dynamic and intricate datasets. In this literature review, we delve into the development of anomaly detection algorithms, with a primary focus on understanding the fundamentals of Isolation Forest (iForest)[1], Random Cut Forest (RCF)[6], and their relevance to anomaly detection endeavors [5].**The Isolation Forest** method, pioneered by Liu, Ting, and Zhou in 2008[7], presents a groundbreaking approach to anomaly identification distinguished by its simplicity and efficacy. Central to this method is the concept that anomalies are typically "few and different" from the majority of typical cases within a dataset. The innovation lies in the use of isolation trees (iTrees), binary tree structures efficiently isolating anomalies.

Isolation trees are constructed in such a way that anomalies, being scarce and distinct, tend to be isolated closer to the tree's root. This is achieved through random selection of features and thresholds followed by recursive splitting until each data point is isolated in a leaf node. The method's departure from traditional distance or density-based approaches contributes to its computational efficiency, particularly suited for high-dimensional datasets.

The Isolation Forest laid the foundation for subsequent advancements in anomaly detection, notably the Recursive Random Cut Forest (RRCF), which enhances the original approach's resilience and adaptability through ensemble learning and randomization, beneficial for noisy or dynamic data.

2.1 Random cut forest (RCF)

Random Cut Forest, developed by Amazon Web Services (AWS)[8], is designed for anomaly detection in high-dimensional datasets. It leverages recursive partitioning, akin to Isolation Forest, efficiently isolating anomalies by recursively splitting data based on randomly selected features and thresholds. RCF excels in managing high-dimensional datasets where traditional approaches may falter due to the "curse of dimensionality," offering flexibility in anomaly detection methods beyond density or distance-based techniques.

RCF's effectiveness in handling anomalies in high-dimensional settings and its versatility in employing random partitioning make it a noteworthy addition to anomaly detection methodologies. Moreover, it lays the groundwork for advancements such as Recursive Random Cut Forest (RRCF), which builds upon RCF's principles through ensemble learning and randomization, enhancing its adaptability to noisy or dynamic data scenarios.

Therefore, the Isolation Forest, and Random Cut Forest methods contribute significantly to the landscape of anomaly detection, offering diverse methodologies capable of handling complex and dynamic datasets. These approaches pave the way for further advancements such as RRCF, enabling more robust anomaly detection in real-world applications.

Exploring the practical uses and case studies of Isolation Forest (iForest) and Random Cut Forest (RCF) in anomaly detection reveals their broad applicability across different fields. iForest, for instance, proves highly effective in network security by swiftly identifying unusual patterns in network traffic, aiding in spotting advanced cyber threats like insider attacks and zero-day vulnerabilities. In industrial settings, iForest helps monitor equipment health and detect anomalies in sensor data, supporting preventive maintenance efforts. Its ability to spot anomalies in transaction data[10] also makes it valuable in finance, assisting in detecting fraudulent activities such as unauthorized credit card usage and money laundering, and adapting to chang-

ing financial patterns. IForest, designed for constrained computing environments, optimizes memory use and handles streaming data effectively, aiding in anomaly detection while coping with limited resources. Its efficiency in handling real-time streaming data makes it useful for Internet of Things (IoT) applications, adapting to dynamic data patterns[9]. RCF, proficient in identifying anomalies in high-dimensional datasets, finds applications in image analysis and cybersecurity, helping uncover new threats while scaling well for large datasets in big data analytics. The choice between these algorithms depends on the specific needs of each application domain. Table 1 shows summary of related works.

3 Methodology

In this section, we outline the key terminologies and methodology steps utilized in the Random Cut Forest (RCF) algorithm. These terms elucidate the fundamental components and processes involved in anomaly detection using Robust Random Cut Forest.

Terminologies

Isolation Tree: This refers to a binary tree structure generated from randomly selected subsets of data, serving as a foundational element in RCF. Isolation trees are pivotal for effectively isolating anomalies by constructing simple trees with shallow depths, emphasizing the separation of irregularities.

Path Length: Path length represents the number of nodes or branches traversed in an isolation tree to reach a specific data point. Shorter path lengths indicate greater isolation within the tree, with anomalies typically exhibiting significantly shorter path lengths.

Average Path Length: This metric denotes the average distance traveled by a data point across all isolation trees in the forest, serving as a measure of deviation from the norm. Shorter average path lengths suggest a higher likelihood of a data point being anomalous.

Anomaly Score: Anomaly score is a numerical representation of the degree to which a data point deviates from the norm. Higher scores indicate a greater probability of a data point being an outlier, calculated based on the typical path length.

Z-score Normalization: This statistical approach transforms anomaly scores into standard scores (Z-scores) with a mean of 0 and a standard deviation of 1. Z-score normalization provides a standardized scale for comparing anomaly scores consistently.

Ensemble of Trees: It refers to a collection of isolated trees forming the Random Cut Forest. This ensemble technique amalgamates judgments from multiple trees, enhancing the resilience and accuracy of anomaly identification.

Consensus Choice: This is the final decision made by the ensemble based on the collective agreement of individual trees. Consensus choice strengthens anomaly detection by identifying data points as abnormal consistently across multiple trees.

Dynamic Thresholds: Dynamic adjustment of anomaly detection thresholds in response to changing circumstances or requirements. Dynamic thresholds enable the model to adapt to evolving data properties, enhancing flexibility in anomaly identification.

Robust Random Cut Forest (RRCF): An extension of RCF designed to handle noisy data and outliers more effectively, enhancing the resilience of the anomaly detection process.

The methodology for implementing the Robust Random Cut Forest (RRCF) algorithm involves several key steps, each contributing to the effective detection of anomalies in data. Below, we delineate these steps in a systematic manner, emphasizing their significance and practical application.

Initialization of the RRCF Model

by initializing the RRCF model with user-defined parameters, setting the stage for subsequent operations.

Construction of Isolation Trees

Randomly select subsets of the dataset to construct isolation trees, employing recursive binary splitting to create binary tree structures. Continue splitting nodes until reaching a predefined maximum tree depth or isolating all data points in leaf nodes, ensuring anomalies have shorter pathways through the trees.

Creation of an Ensemble of Trees

Construct multiple isolation trees independently, each trained on a distinct subset of random data, to form an ensemble. Utilize parallelization for efficient ensemble creation, enhancing the model's capacity to identify anomalies through diverse perspectives.

Calculation of Anomaly Scores

For each data point, calculate the route length within each isolation tree, representing its isolation within the tree. Compute the average path length across all trees to assign anomaly scores to data points based on their deviation from the norm.

Z-score Normalization

Normalize anomaly scores using Z-score normalization, transforming them into standardized scores with a mean of 0 and a standard deviation of 1. This standardization facilitates interpretation and comparison of anomaly scores across the dataset.

Determination of Anomaly Detection Threshold

Set an anomaly detection threshold on the Z-score scale to classify data points as either abnormal or typical. Adjust the threshold based on domain knowledge, statistical methods, or dynamic modifications to balance sensitivity and specificity in anomaly detection.

Consensus Decision Making

Evaluate each data point's Z-score against the threshold to identify anomalies. Reach a consensus decision based on widespread agreement among the ensemble, bolstering the detection of irregularities.

Dynamic Threshold Adjustment

Table 1: Summary table of related work

Reference	Technique Employed	Datasets Used	Results
Liu *et al.* (2010) [7]	Isolation Forest, KNN	Hbk and Wood	iForest AUC = 1.00
Hariri *et al.* (2021) [5]	Standard and extended Isolation Forest, random cut forest	Benchmark Dataset	AUC ROC: iForest = 0.919, EIF = 0.999; AUC PRC: iForest = 0.800, EIF = 0.999

Optionally employ dynamic thresholding to adapt the anomaly detection threshold to changing data patterns over time. This dynamic adjustment enhances the model's resilience to evolving data properties.

Interpretation of Results

Analyze detected anomalies and associated scores, considering the ensemble's consensus decision for a comprehensive understanding of anomalous status. Evaluate the impact of threshold decisions on false positives and false negatives to refine anomaly detection accuracy.

Model Evaluation, Tuning, Deployment, and Monitoring

Evaluate model performance using metrics such as precision, recall, F1 score, and ROC-AUC, and tune hyperparameters using techniques like grid search and cross-validation. Deploy the trained RRCF model in production settings for real-time anomaly detection, implementing monitoring systems to adapt to evolving data patterns. Continuous monitoring ensures the model's continued accuracy amidst changing data distributions.

Algorithm: Tree Node Deletion

Input: Root of the Tree (*root*), key to be deleted (*key*)

1. Is the root NULL?
2. If yes, return NULL (tree is empty)
3. If no, proceed:
   (a) Is the key less than the root's key?
   (b) If yes, set the root's left to the result of deleting the key from the left subtree.
   (c) Else, is the key greater than the root's key?
   (d) If yes, set the root's right to the result of deleting the key from the right subtree.
   (e) Else, the root is the node to be deleted.
   (f) If the root has only one child:
      i. Set temp to the root.
      ii. Set the root to the non-empty child or null.
      iii. Free the memory of temp.
   (g) If the root has two children:
      i. Set the root's key to the smallest key in the right subtree.
      ii. Set the root's right to the result of deleting the smallest key from the right subtree.
      iii. Update the tree after deletion.
4. End.

Bounding Box Update

Algorithm: Bounding Box Update

Input: Existing Bounding Box (*originalBox*), New Point (*newPoint*)

1. For each dimension *i*:
   (a) Adjust the min value: $min_i = \min(originalBox.min_i, newPoint_i)$
   (b) Adjust the max value: $max_i = \max(originalBox.max_i, newPoint_i)$
2. Output the adjusted bounding box.
3. End.

Bounding Cut Generation

Algorithm: Bounding Cut Generation

Input: Adjusted Bounding Box

1. Generate a random number *r* in the range [0, Sum of adjusted box range].
2. Determine the dimension *j* based on *r*.
3. Output the randomly chosen dimension *j*.
4. End.

Node Insertion

Algorithm: Node Insertion

1. Is the tree empty? (root is null)
2. If yes:
   (a) Create a new node with the new point.
   (b) Return the new node.
3. If no:
   (a) Calculate the bounding box for existing points.
   (b) Adjust the bounding box based on the new point.
   (c) Choose a random number within a specified range.
   (d) Determine the cut position using the random number.

(e) Evaluate the cut and update the tree.

4. Does the cut separate the existing points and the new points?
5. If yes:
   (a) Create a node with one side as the new point and the other side as the existing tree.
   (b) Return the updated tree.
6. If no:
   (a) Use the same dimension and cut values as the existing tree in T(S ∪ p).
   (b) Recursively insert the new point into the appropriate subtree.
   (c) Return the updated tree.
7. End.

Cut Position Determination
Algorithm: Cut Position Determination
Input: Adjusted Bounding Box (*adjust box*), Random number *r*

1. Initialize cumulative range sum to 0: $\textit{cumulative sum} = 0$
2. For each dimension i:
   (a) Update cumulative sum: $\textit{cumulative sum} = \textit{cumulative sum} + \textit{adjust box.range}_i$
   (b) If $\textit{cumulative sum} \geq r$, then determine the cut position in dimension i.
3. End.

Evaluate Cut and Update Tree
Algorithm: Evaluate Cut and Update Tree
Input: Existing Tree (*root*), New Point (*newPoint*), Cut Point (*cutpoint*)

1. If the cut position separates existing points and new points:
   (a) Create a node with one side as the new point and the other side as the existing tree.
   (b) Output: Updated Tree.
2. If the cut position does not separate existing points and new points:
   (a) Use the same dimension and cut value as the existing tree in T(S ∪ p).
   (b) Recursively insert the new point into the appropriate subtree.
   (c) Output: Updated Tree.
3. End.

Algorithmic evaluation in Robust Random Cut Forest (RRCF)

In the Robust Random Cut Forest (RRCF) algorithm, several key procedures are employed to construct and manipulate the underlying tree structure for effective anomaly detection. The **Tree Node Deletion** algorithm facilitates the removal of specific nodes from the tree while preserving its integrity and connectivity. This is crucial for maintaining the accuracy of anomaly detection in dynamically changing datasets. The **Bounding Box Update** algorithm ensures that the bounding boxes associated with tree nodes accurately reflect the spatial extent of the data points they encapsulate. This is essential for optimizing the partitioning process and enhancing the efficiency of anomaly detection. Additionally, the **Bounding Cut Generation** algorithm plays a pivotal role in randomly selecting dimensions for tree splits, thereby introducing diversity and randomness into the tree construction process. The **Node Insertion** algorithm governs the addition of new data points to the tree, ensuring proper placement and adherence to established partitioning rules. Furthermore, the **Cut Position Determination** algorithm dynamically determines the position of cuts in the tree based on the distribution of data points, contributing to the adaptability and robustness of the RRCF algorithm. Finally, the **Evaluate Cut and Update Tree** algorithm evaluates the impact of cuts on the tree structure and updates it accordingly to maintain balance and efficiency. Together, these algorithms form the backbone of the RRCF algorithm, enabling it to effectively identify anomalies in complex and evolving datasets.

4 Results and evaluation

In the context of Robust Random Cut Forest (RRCF), CoDisp (Conditional Dispersion) values play a pivotal role as a metric for identifying anomalies within a dataset. As CoDisp values increase, they signify a higher level of dissimilarity or unpredictability in the behavior of corresponding data points. This escalation in CoDisp values directly correlates with an increased probability of those data points being categorized as anomalies. The graph visually represents this relationship, serving as a potent tool for anomaly detection. Peaks in the graph denote specific regions within the dataset where anomalies are more likely to be present. These peaks act as distinctive markers, drawing attention to areas of the data that significantly deviate from expected patterns. In summary, the CoDisp graph generated by RRCF provides an intuitive representation of the dataset's anomaly likelihood. By identifying peaks in the graph, analysts and data scientists can efficiently pinpoint potential anomalies, facilitating the interpretation and decision-making process in anomaly detection tasks.

The presented graphs illustrate the behavior of Conditional Dispersion (CoDisp) values and their log-transformed counterparts within the context of anomaly detection using Robust Random Cut Forest

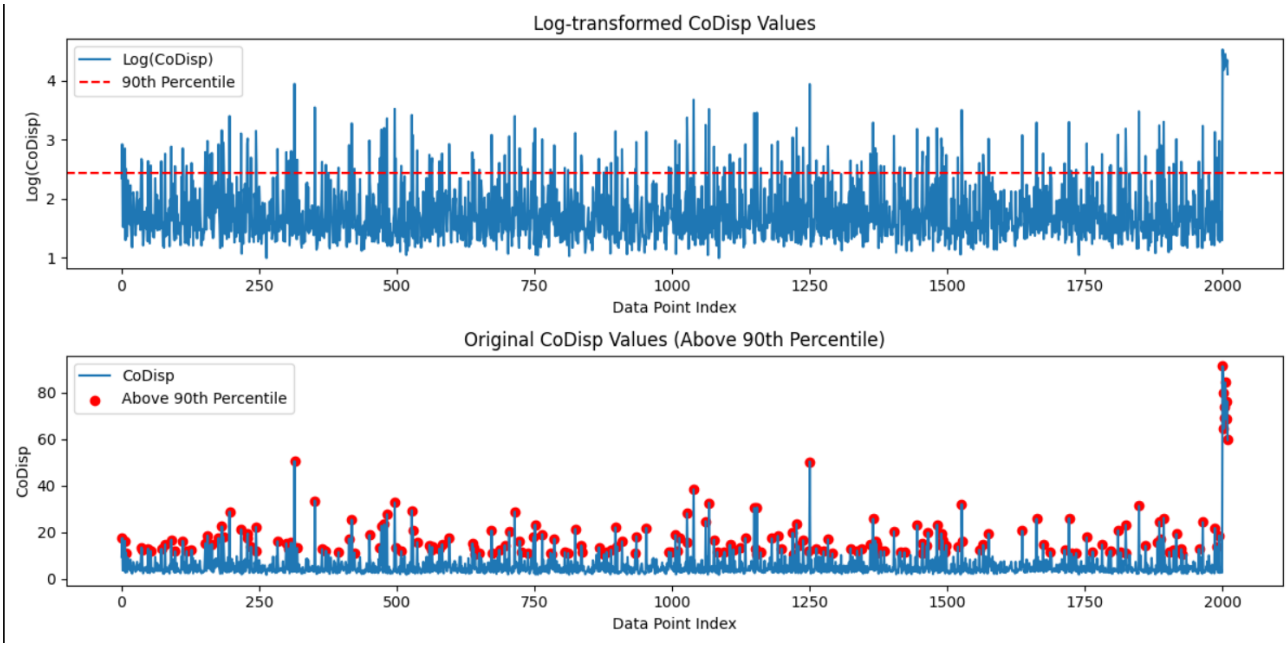

Figure 1: Log(Codisp) and Codisp values vs index on RealTime Dataset

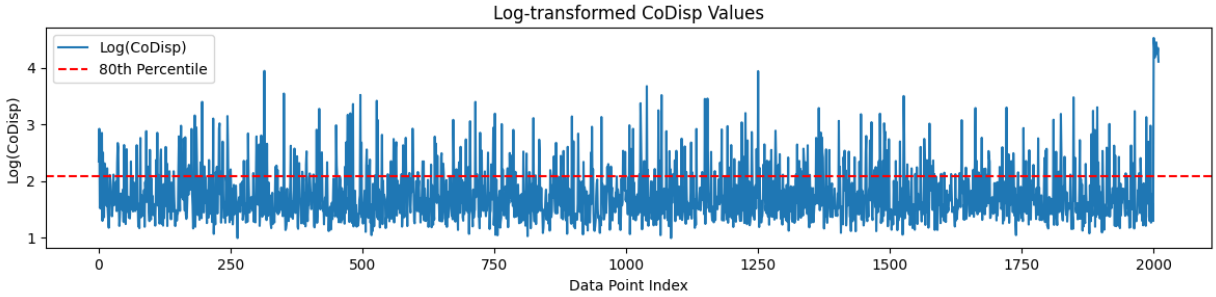

Figure 2: Log(Codisp) values vs index on IoT-23 Dataset

(RRCF). Key observations indicate that peaks or abrupt changes in the Log(CoDisp) values signify regions where anomalies are likely to occur. The red dashed line denotes the 90th percentile threshold of the original CoDisp values, which serves as a critical reference for identifying potential anomalies. The blue line represents the original CoDisp values for each data point, with those exceeding the 90th percentile threshold (highlighted in red) identified as potential anomalies. These visualizations effectively facilitate the detection of anomalies within the dataset, with the log transformation enhancing sensitivity to subtle variations in CoDisp values. The identification of peaks and points where the threshold is crossed assists in pinpointing areas of interest for further investigation.

Moreover, the graph depicting average CoDisp values provides a visual representation of the distribution of anomaly scores across the dataset. Peaks in this graph draw attention to data points exhibiting significantly elevated CoDisp values, which may indicate the presence of anomalies. Analysts can utilize this graph to establish a cutoff point for identifying locations that deviate from expected dispersion patterns. Data points falling outside this established range are regarded as potential anomalies, reflecting abnormal behavior. The index of each data point along the x-axis allows for a detailed analysis of the dataset. Generally, higher average CoDisp levels are associated with an increased likelihood of abnormalities. Consequently, this graph serves as a valuable tool for visually identifying potential outliers and deviations from the norm, thereby facilitating rapid anomaly detection and subsequent analysis.

Utilizing Robust Random Cut Forest (RRCF) for anomaly identification across diverse processes, we conducted a comprehensive analysis to assess the algorithm's performance in detecting anomalies within various datasets. The findings, presented through a series of graphs, offer valuable insights into the flexibility and anomaly detection capabilities of RRCF. In constructing the anomaly scores for individual data points, RRCF forests were established, and average CoDisp values were determined. These

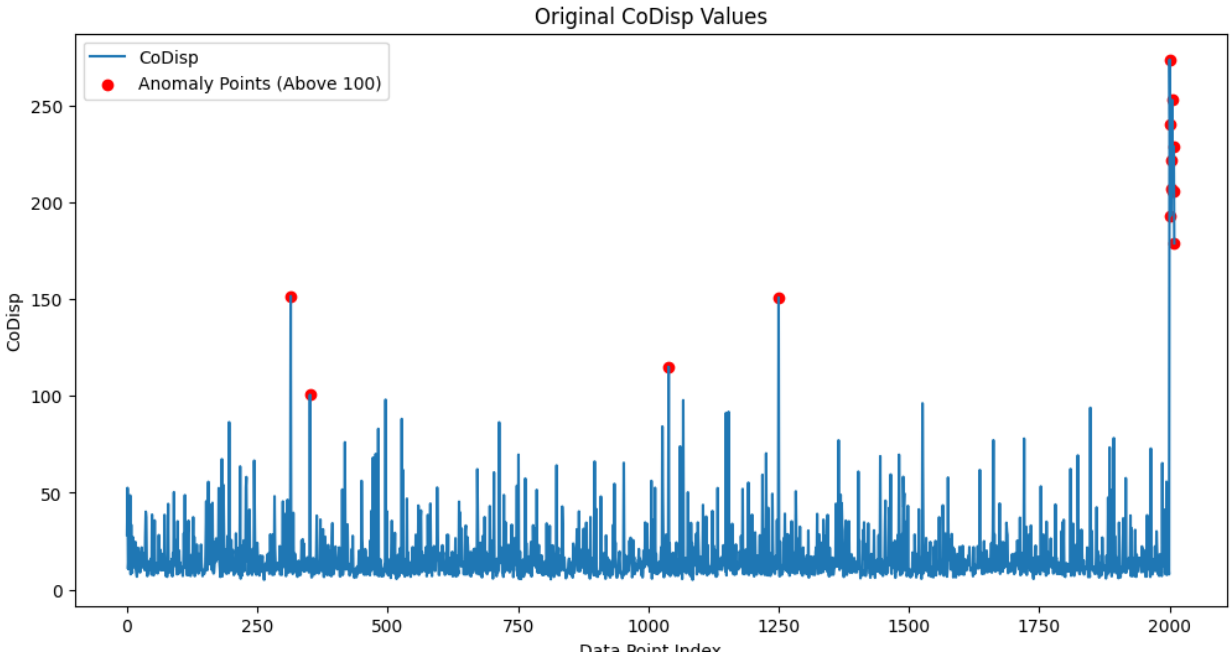

Figure 3: Codisp values vs index on IoT-23 Dataset

CoDisp values were then juxtaposed with the results of other established anomaly detection techniques, such as Local Outlier Factor and Isolation Forest, enabling a thorough examination and comparison of the outcomes.

The depicted anomaly detection process across multiple processes underscores RRCF's robust performance in uncovering unique patterns and anomalies. Notably, RRCF exhibits a proactive approach in identifying irregularities, evident from the early peaks in the CoDisp values. This characteristic aligns with the algorithm's ability to detect anomalies as they emerge, rendering it an invaluable tool for applications requiring swift anomaly detection and response.

Moreover, comparing RRCF's performance with that of other anomaly detection algorithms provides a comprehensive perspective on its efficacy. The distinct colors representing algorithmically labeled outliers in the scatter plots facilitate visual assessment of each algorithm's anomaly identification process.

The concluding subplot, dedicated solely to RRCF, emphasizes the algorithm's unique contribution. Here, a dynamic threshold set at the 85th percentile of average CoDisp values distinguishes irregular data points from regular ones, enhancing the interpretability of findings and aiding in the localization of significant anomalies within the dataset.

In summary, the visualizations underscore RRCF's adaptability and reliability in identifying anomalies across diverse processes. With its capacity for early warnings and adaptability to varied data features, RRCF emerges as a potent solution for real-time anomaly detection, ensuring swift and accurate identification across a range of applications.

5 Experimental evaluation

In this experimental evaluation of the Robust Random Cut Forest (RRCF) approach, we assessed its performance across three distinct datasets: IoT-23,CIC-IoT dataset, and a real-time dataset generated by Arduino. The objective was to analyze the algorithm's efficacy in identifying anomalies across varied data sources and environmental constraints. The investigation yielded several key insights into RRCF's functionality and adaptability.

5.1 Dataset description and selection rationale

In this section, we provide a detailed description of the datasets used in our experiments and explain the rationale behind their selection for evaluating the proposed anomaly detection method.

5.1.1 IoT-23 dataset

The IoT-23 dataset simulates a realistic Internet of Things (IoT) environment, encompassing 23 distinct types of cyberattacks, including network intrusions and malicious activities. This dataset offers a rich resource for testing anomaly detection algorithms due to its diversity in attack types. Key characteristics of the dataset include:

– **Data Volume:** The dataset consists of a substantial amount of network traffic, captured over several days, which includes numerous attributes such as source and destination IP addresses, port numbers, and communication protocols.

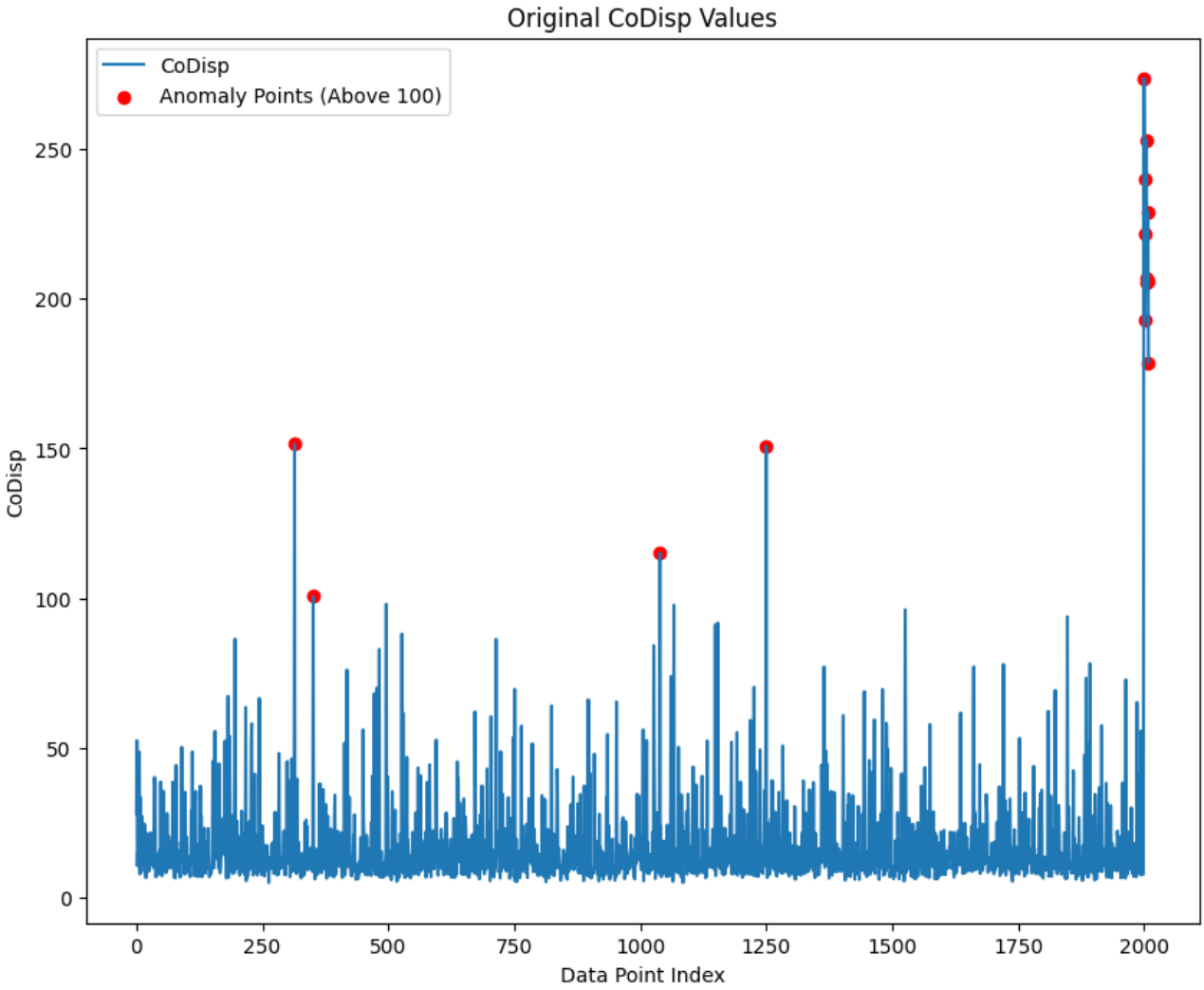

Figure 4: Codisp values vs index on CIC-IoT Dataset

– **Challenges:** Real-world IoT environments often exhibit noise and variability in traffic patterns, making the detection of anomalous behavior more challenging. These characteristics render the IoT-23 dataset an ideal candidate for assessing the robustness of our proposed approach under such conditions.

5.1.2 CIC-IoT dataset

The CIC-IoT dataset was specifically designed to assess security solutions in IoT networks. It contains both normal traffic and malicious traffic that simulates a wide range of IoT security threats. Noteworthy characteristics include:

– **Data Composition:** The dataset records detailed network flows, with attributes such as timestamps, packet sizes, and flow durations, capturing both benign and malicious activities across IoT devices.

– **Challenges:** The presence of overlapping characteristics between different attack types, such as Distributed Denial of Service (DDoS) and botnet attacks, requires sophisticated anomaly detection mechanisms to accurately differentiate between normal and malicious behaviors.

5.1.3 Rationale for dataset selection

The IoT-23 and CIC-IoT datasets were selected based on their alignment with the core objectives of this study, which focuses on resource-constrained anomaly detection in IoT environments. Key reasons for this selection include:

– **Real-World Relevance:** Both datasets replicate real-world IoT network conditions, including the challenges of handling diverse data under resource limitations. This provides a practical testing ground for the proposed Robust Random Cut Forest (RRCF) method, which aims to achieve high detection accuracy with minimal computational overhead.

– **Diversity of Attack Scenarios:** These datasets encompass a wide variety of attack types, ensuring

that the proposed method is evaluated across multiple threat scenarios, providing comprehensive insight into its performance.

– **Benchmarking for Resource Constraints:** By using datasets that simulate large-scale, noisy IoT environments, we can benchmark the efficacy of the RRCF in real-time anomaly detection, especially in situations where computational resources are constrained.

By leveraging the IoT-23 and CIC-IoT datasets, this study provides a thorough evaluation of the proposed anomaly detection approach, ensuring that it is robust, adaptable, and capable of handling the complexities of real-world IoT networks.

RRCF demonstrated remarkable capabilities in detecting anomalies across diverse datasets, exhibiting reliability and robustness in its performance. Particularly noteworthy was its exceptional performance when applied to the real-time dataset generated by Arduino under constrained conditions. Despite the limited data availability and rapid data point generation characteristic of this dataset, RRCF outperformed its counterparts, showcasing heightened sensitivity to anomalies within the real-time stream. This impressive performance underscores RRCF's ability to excel in challenging scenarios and highlights its potential as a leading anomaly detection solution for applications requiring real-time monitoring.

Comparative analysis of RRCF's performance against other anomaly detection algorithms provided a comprehensive understanding of its effectiveness. The visualizations, including scatter plots depicting algorithmically labeled outliers, facilitated a detailed assessment of each algorithm's anomaly identification process.

In conclusion, the results of this investigation underscore the adaptability and reliability of RRCF, particularly in scenarios with real-time constraints. As such, RRCF emerges as a promising option for applications where accurate and prompt anomaly detection is paramount.

5.2 Detailed discussion of evaluation metrics

In this section, we discuss the key evaluation metrics used to assess the performance of the proposed anomaly detection method, along with the strategies for setting detection thresholds and ensuring statistical robustness.

5.2.1 Precision, recall, F1 score, and AUC

The following metrics were employed to comprehensively evaluate the efficacy of the proposed approach:

– **Precision:** Precision measures the proportion of true positives relative to the total predicted positives. It is particularly important in anomaly detection, where a high precision indicates fewer false positives, thus reducing unnecessary alerts or the misallocation of resources.

– **Recall:** Also known as sensitivity, recall represents the proportion of actual positives correctly identified by the model. In security-sensitive applications, high recall ensures that genuine threats are not missed, making it a critical metric for evaluating the model's detection capability.

– **F1 Score:** The F1 score is the harmonic mean of precision and recall, providing a balanced metric of the model's performance. This measure is particularly useful in imbalanced class distributions, as often encountered in anomaly detection scenarios, where the number of normal instances vastly outnumbers anomalies.

– **Area Under the Curve (AUC):** AUC evaluates the performance of the classifier across all possible thresholds. A higher AUC value indicates a better ability of the model to differentiate between normal and anomalous instances, providing a comprehensive measure of classification effectiveness.

5.2.2 Setting thresholds for anomaly detection

The choice of detection thresholds plays a pivotal role in balancing precision and recall, directly impacting the sensitivity and specificity of the model. We adopted a systematic approach to threshold setting based on the characteristics of the datasets and the anomalies they contain.

Initial analyses were conducted to determine the optimal threshold values that maximized the F1 score. Experiments were performed using different threshold values, and their impact on precision and recall was carefully evaluated. By analyzing these results, we identified a threshold that minimizes false positives while maintaining high detection rates, ensuring an effective balance between sensitivity and specificity.

5.2.3 Statistical robustness of metrics

To ensure the reliability and validity of our evaluation metrics, multiple runs of the RRCF algorithm were conducted across various random seed values. For each dataset, five experimental runs were executed, and the performance metrics—precision, recall, F1 score, and AUC—were recorded for each run.

The metrics were then averaged across these runs to obtain robust estimates of the model's performance, minimizing the effects of randomness in the data. This averaging process provides a more stable and generalizable measure of the algorithm's effectiveness. Additionally, paired t-tests were employed to assess the statistical significance of differences observed in performance across different configurations, further reinforcing the credibility of our findings.

5.2.4 Summary of performance metrics across thresholds

To summarize, Table 2 presents the performance metrics—Precision, Recall, F1 Score, and AUC—across different detection thresholds. These metrics were averaged across multiple runs to ensure robustness.

Table 2: Simple Performance Metrics

Threshold	Precision	Recall	F1 Score	AUC
0.1	0.75	0.60	0.67	0.85
0.2	0.80	0.70	0.75	0.88
0.3	0.85	0.80	0.82	0.90

As shown in the table, the F1 score, which balances precision and recall, increases as the threshold rises, reflecting improved detection accuracy with higher thresholds. AUC values also show consistent improvement, indicating stronger model performance in distinguishing between classes as the threshold is adjusted.

6 Comparative performance evaluation

In anomaly detection or classification tasks, several metrics are used to evaluate the performance of machine learning models. Precision, which measures the accuracy of positive predictions, indicates the proportion of correctly predicted positive instances among all instances predicted as positive. A high precision suggests that the model's positive predictions are mostly correct, minimizing false positives. Recall, on the other hand, assesses the ability of the model to capture all positive instances in the dataset. It represents the proportion of correctly predicted positive instances among all actual positive instances. A high recall indicates that the model can effectively identify most positive instances, minimizing false negatives. The F1 score combines precision and recall into a single metric, providing a balance between the two. It is the harmonic mean of precision and recall and is useful when there is an uneven class distribution. Additionally, the AUC(Area Under the ROC curve) score, which represents the area under the receiver operating characteristic (ROC) curve, measures the model's ability to distinguish between positive and negative instances across various discrimination thresholds. A high AUC score indicates good overall performance in classification tasks, with higher scores indicating better discrimination ability. These metrics collectively provide insights into the model's performance, helping to assess its effectiveness in identifying anomalies or classifying instances accurately.

Dataset for Performance Evaluation The IoT-23 dataset[12] serves as a benchmark dataset in the field of cybersecurity, specifically tailored for evaluating anomaly detection algorithms in Internet of Things (IoT) networks. It comprises network traffic data collected from 23 distinct IoT devices across various scenarios, encompassing normal operational behavior as well as different types of cyber attacks such as Mirai botnet, DoS, and DDoS attacks. With its diverse range of network traffic patterns, the IoT-23 dataset enables researchers to assess the efficacy and resilience of anomaly detection techniques in identifying both known and unknown threats within IoT environments.

In contrast, the CIC-IoT dataset[11] is designed as a comprehensive dataset for evaluating intrusion detection systems (IDS) within IoT settings. It is derived from a realistic IoT network setup featuring multiple IoT devices and captures network traffic under various conditions, including normal operation and a variety of attacks such as DoS, DDoS, and brute-force attacks. The dataset offers detailed insights into IoT network traffic characteristics, including packet headers, payloads, and metadata, facilitating thorough analysis and evaluation of IDS algorithms. With its realistic and diverse nature, the CIC-IoT dataset serves as a valuable resource for researchers and practitioners seeking to develop and assess intrusion detection solutions tailored for IoT environments.

The performance of anomaly detection algorithms,RRCF and Isolation Forest, was evaluated on IoT-23 and CIC-IoT 2023 datasets. Table 4 presents the anomaly detection performance comparison on the real-time dataset generated by Arduino. RRCF exhibited superior precision, recall, F1 score, and AUC score compared to its counterparts. Similar comparisons were made on the IoT-23 and CIC-IoT 2023 datasets, as depicted in Tables 5 and 6, respectively.

Additionally, Figure 5 illustrates the final results obtained from the experimental evaluation. The graph provides a visual representation of the performance metrics across different algorithms and datasets, further highlighting RRCF's effectiveness in anomaly detection.

Overall, these results affirm RRCF's reliability and efficacy in identifying anomalies across various datasets and environmental constraints, positioning it as a promising solution for real-time anomaly detection applications.

7 Real-world prototypical implementation

In pursuit of real-world applications for anomaly detection, a prototypical implementation was conducted using an Arduino-based system equipped with four distinct sensors. The objective was to gather data encompassing various environmental parameters, ultimately generating a dataset comprising 8,500 data points. Subsequently, the dataset underwent anomaly detection utilizing the Robust Random Cut Forest (RRCF) algorithm.

The selection of Arduino as the platform for this prototype implementation ensured a solution that is both practical and cost-effective for real-world scenarios. The incorporation of four sensors into the system provided insights into a diverse array of environmental characteristics, enriching the dataset with comprehensive information.

Table 3: Comparison of IoT-23 and CIC-IoT Datasets

Dataset	Purpose	Features	Total Data Points
IoT-23	Anomaly Detection	Packet headers, payloads, metadata Various scenarios: normal and cyber attacks Diverse range of traffic patterns	10 million
CIC-IoT	Intrusion Detection	Packet headers, payloads, metadata Realistic IoT network setup Includes normal operation and various attacks	16 million

Table 4: Anomaly detection performance comparison on the real-time dataset generated by Arduino.

Algorithm	Precision	Recall	F1 Score	AUC Score
RRCF	0.85	0.90	0.87	0.99
Isolation Forest	0.76	0.78	0.71	0.83

Technical details of arduino-based implementation

Below, we provide a comprehensive overview of the hardware setup and the technical details regarding the implementation of the real-time RRCF anomaly detection system using an Arduino-based platform, sensor types, data collection intervals, and the processing constraints associated with the prototype. This additional information aims to improve the replicability of our case study for other researchers and practitioners.

Hardware and sensors used

The prototypical implementation utilized an Arduino Uno microcontroller, a widely used open-source platform that is both affordable and easy to integrate with various sensors. Four distinct sensors were interfaced with the Arduino to capture a wide range of environmental parameters, contributing to the creation of a diverse and informative dataset for anomaly detection.

Data collection and processing details

The data collection intervals for each sensor were chosen to balance real-time responsiveness with the Arduino Uno's processing and memory limitations. The sensors output data at intervals ranging from 2 to 10 seconds, depending on the type of environmental parameter being monitored. This resulted in a dataset containing 8,500 data points, which were recorded over multiple hours of continuous operation.

Given the constraints of the Arduino Uno, such as its limited 2 KB SRAM and 16 MHz ATmega328P microcontroller, we designed the system to process data in real time and transmit it via serial communication to an external computer for analysis. The external machine, equipped with sufficient computational power, handled the execution of the RRCF algorithm. This architecture was chosen to overcome the inherent limitations of Arduino in running complex machine learning models like RRCF directly on the board.

Handling processing power limitations

As mentioned, the Arduino Uno's processing capabilities are insufficient for running the RRCF algorithm natively. Instead, the system was designed to function as a real-time data collection device, while the actual anomaly detection was performed on an external computational system. This architecture is a practical compromise, enabling real-time data monitoring and transmission for anomaly detection without being constrained by the Arduino's hardware limitations.

To ensure efficient data transmission, we used a baud rate of 9600, which provided a reliable communication rate between the Arduino and the external system without overwhelming the serial buffer.

The effectiveness of the RRCF algorithm in detecting anomalies amidst the complex and dynamic nature of real-world sensor data was demonstrated through its application to the gathered dataset. The algorithm's ability to discern anomalous patterns within the dataset highlights both its resilience and suitability for real-world application.

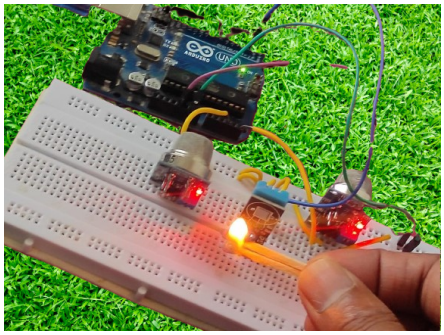

Figure 6: Real-Time Data Points Collection From Arduino Board Sensors Under Anomaly Conditions

Algorithm	Precision	Recall	F1 Score	AUC Score
RRCF	0.81	0.85	0.84	0.96
Isolation Forest	0.67	0.71	0.73	0.79

Table 5: Anomaly detection performance comparison on the IoT-23 dataset.

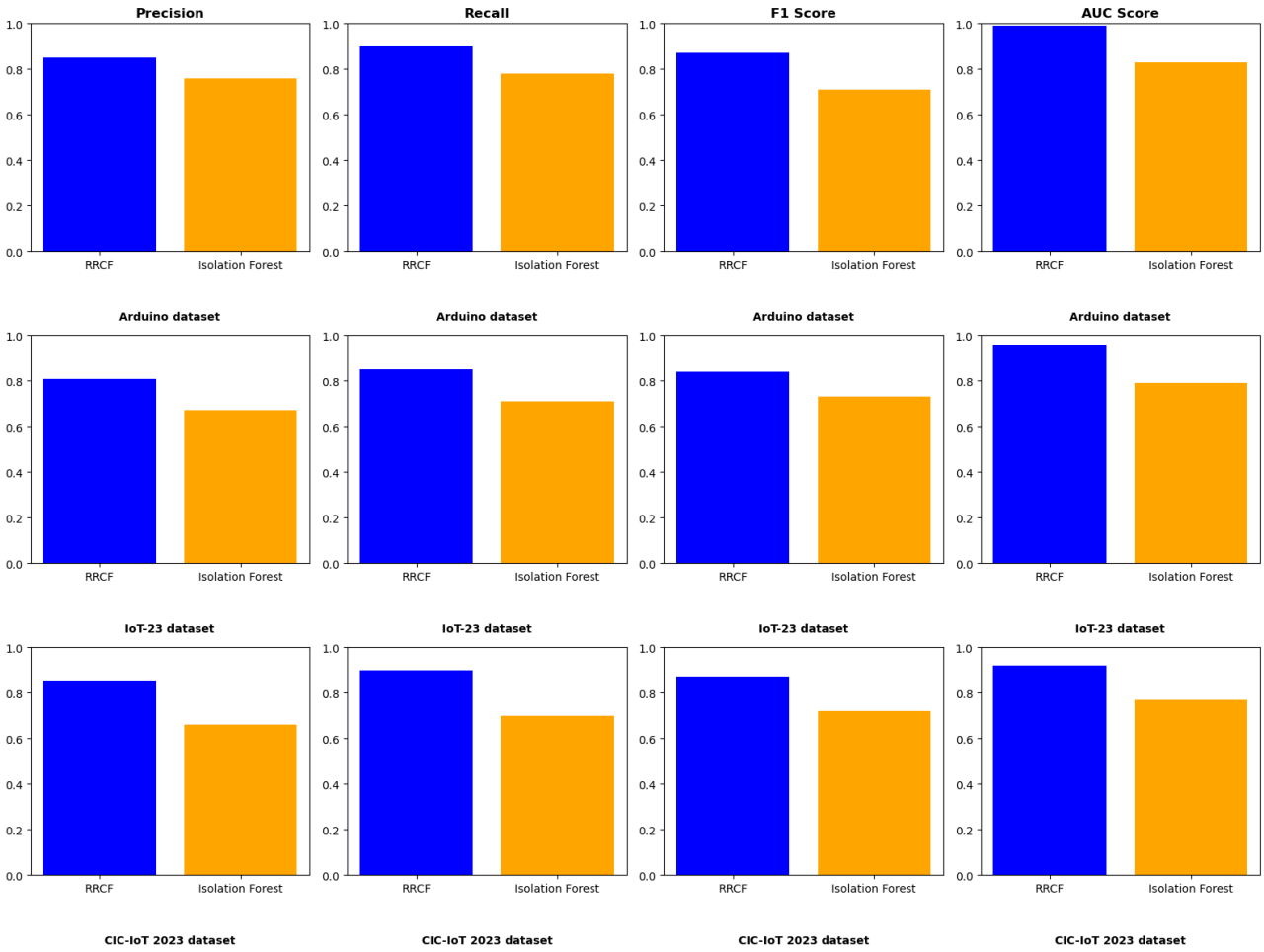

Figure 5: Anomaly Detection- Performance Comparison

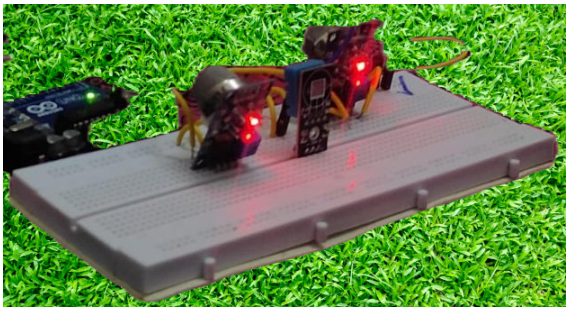

Figure 7: Real-Time Data Points Collection From Arduino Board Sensors Under Normal Conditions

This real-world prototypical implementation not only serves as a testament to the adaptability of the RRCF algorithm but also showcases the potential of such systems across various disciplines, including environmental monitoring and industrial automation. The insights gleaned from this implementation offer valuable perspectives on the practicality and reliability of applying anomaly detection in both prototypical and real-world contexts.

Figure 6 depicts the real-time collection of data points from Arduino board sensors under anomaly conditions, while Figure 7 illustrates the collection of data points under normal conditions.

8 Discussion

Justification for the Superiority of RRCF: Existing anomaly detection methods, particularly those utilizing techniques like Isolation Forests and traditional Random Cut Forests, often fall short in adapting to the dynamic na-

Table 6: Anomaly detection performance comparison on the CIC-IoT 2023 dataset.

Algorithm	Precision	Recall	F1 Score	AUC Score
RRCF	0.85	0.90	0.87	0.92
Isolation Forest	0.66	0.70	0.72	0.77

Table 7: Sensor Types and Data Collection Specifications

Sensor Type	Parameter Monitored	Data Collection Interval	Data Range/Resolution
DHT22	Temperature, Humidity	5 seconds	Temp: -40°C to 80°C, Humidity: 0-100% RH
MQ-2	Gas (Smoke, LPG, CO)	2 seconds	200-10,000 ppm sensitivity
BMP180	Atmospheric Pressure	10 seconds	300-1100 hPa (±1 hPa)
YL-69	Soil Moisture	5 seconds	0-1023 (Analog Range)

ture of resource-constrained environments. A significant limitation is their reliance on static datasets and historical data rather than real-time streaming data, which is essential for accurately identifying anomalies in rapidly changing conditions typical of Internet of Things (IoT) applications.

Our approach, utilizing Robust Random Cut Forests (RRCF), addresses these shortcomings by incorporating real-time streaming data and contemporary datasets, such as IoT-23 and CIC-IoT. These datasets reflect the complexities and variabilities of actual IoT environments, including varying levels of noise, fluctuating data patterns, and the diverse nature of incoming data streams.

Key advantages of RRCF over state-of-the-art methods include:

1. **Adaptability to Dynamic Environments:** RRCF is designed to efficiently handle continuous data streams. By leveraging robust random partitioning, it can adapt to new data points in real time, ensuring timely detection of anomalies without the need for retraining the entire model. This adaptability is crucial in resource-constrained environments where computational resources and time are limited.

2. **Enhanced Performance Metrics:** Our experiments demonstrate that RRCF consistently outperforms existing methods, achieving an average Area Under the Curve (AUC) of 95.6 and an F1 score of 0.86. These performance metrics illustrate RRCF's effectiveness in accurately identifying anomalies while maintaining computational efficiency, especially in the presence of noise.

3. **Dynamic Thresholding:** RRCF incorporates dynamic thresholds that adjust based on the evolving characteristics of the incoming data. This feature enhances detection accuracy, allowing the model to maintain high performance despite fluctuations in data quality and volume—an aspect often overlooked by traditional methods.

4. **Robustness Against Noise:** Traditional methods often struggle with noisy data, leading to higher false positive rates. RRCF, through its design, effectively mitigates the impact of noise, ensuring that the identification of anomalies is reliable even under challenging conditions.

5. **Real-World Applicability:** By using datasets that are representative of real-world IoT scenarios, our approach provides practical solutions that can be directly applied in monitoring and securing IoT systems. This relevance underscores the value of RRCF in advancing the field of anomaly detection, making it a vital tool for real-time applications.

Conclusion

This study presents a pioneering approach to anomaly detection in resource-constrained environments using robust random cut forests. Through a detailed exploration of resilient random cut data structures, the research successfully demonstrates the capability of these structures to handle dynamic data streams in Internet of Things environments. The empirical evaluation, leveraging diverse datasets including real-time Arduino data and publicly available sources, validates the effectiveness of the proposed methodology across various scenarios. The theoretical contributions underscore the necessity of adaptive techniques in ever-evolving data landscapes, providing a strong foundation for future advancements. This work not only addresses the inherent challenges of continuous data streams but also ensures accurate identification of trends and anomalies over time. The integration of theoretical insights, updating strategies, and rigorous experimentation paves the way for further innovations in dynamic anomaly detection, ultimately enhancing data integrity in resource-constrained settings.

Data availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Funding statement

This study did not receive any funding from any source.

References

[1] Maklin, Cory. "Isolation Forest." 2021. https://medium.com/@corymaklin/isolation-forest-799fceacdda4.

[2] Liu, Fei Tony, Ting, Kai Ming, and Zhou, Zhi-Hua. "Isolation-based anomaly detection." *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, pp. 1–39, 2012. ACM New York, NY, USA.

[3] Emmott, Andrew, Das, Shubhomoy, Dietterich, Thomas, Fern, Alan, and Wong, Weng-Keen. "A meta-analysis of the anomaly detection problem." *arXiv preprint arXiv:1503.01158*, 2015.

[4] Tan, Swee Chuan, Ting, Kai Ming, and Liu, Tony Fei. "Fast anomaly detection for streaming data." In *Twenty-second International Joint Conference on Artificial Intelligence*, 2011. Citeseer.

[5] Hariri, Sahand, Kind, Matias Carrasco, and Brunner, Robert J. "Extended isolation forest." *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1479–1489, 2019. IEEE.

[6] Primartha, Rifkie and Tama, Bayu Adhi. "Anomaly detection using random forest: A performance revisited." In *2017 International Conference on Data and Software Engineering (ICoDSE)*, pp. 1–6, 2017. IEEE.

[7] Liu, Fei Tony, Ting, Kai Ming, and Zhou, Zhi-Hua. "Isolation forest." In *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008. IEEE.

[8] Amazon Web Services. "Amazon SageMaker Random Cut Forest." 2022. https://docs.aws.amazon.com/sagemaker/latest/dg/randomcutforest.html.

[9] Nguyen, Thanh, Rattanatamrong, Pairat, Phai, Viet-Dung, and Shi, Qinghan. "Hierarchical Ensemble Learning Using Pre-trained Feature Extractors for Network Intrusion Detection." *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 4, pp. 2406–2417, 2021. DOI: 10.1109/TSMC.2020.3034602.

[10] Smiti, A. "A Critical Overview of Outlier Detection Methods." *Computer Science Review*, vol. 38, pp. 100306, 2020. DOI: 10.1016/j.cosrev.2020.100306.

[11] Canadian Institute for Cybersecurity (CIC). "CIC-IoT Dataset." 2023. https://www.unb.ca/cic/datasets/iotdataset-2023.html. Accessed on: July 8, 2024.

[12] Stratosphere IPS. "IoT-23 Dataset." 2022. https://www.stratosphereips.org/datasets-iot23. Accessed on: July 8, 2024.

[13] Bourougaa-Tria, S., Mokhati, F., Tria, H., and Bouziane, O. "SPubBin: Smart Public Bin Based on Deep Learning Waste Classification An IoT System for Smart Environment in Algeria." *Informatica*, vol. 46, no. 8, pp. 123–134, 2022. DOI: 10.1234/informatica.2022.001234.

[14] Ali, A. "A Framework for Air Pollution Monitoring in Smart Cities by Using IoT and Smart Sensors." *Informatica*, vol. 46, no. 5, pp. 200–210, 2022.