Large-Scale Curated Multivariate Time Series Anomaly Detection Dataset for Laptop Performance Metrics

Veena More*, Sowmya Kella*, Ramesh K*
Karnataka State Akkamahadevi Women's University, State Highway 12
Athani Rd, Jnanashakti campus, Torvi, Vijayapura, 586108
Karnataka, India

Abstract

High-quality multivariate time-series datasets are significantly less accessible compared to more common data types such as images or text, due to the resource-intensive process of continuous monitoring, precise annotation, and long-term observation. This paper introduces a cost-effective solution in the form of a large-scale, curated dataset specifically designed for anomaly detection in computing systems' performance metrics. The dataset encompasses 45 GB of multivariate time-series data collected from 66 systems, capturing key performance indicators such as CPU usage, memory consumption, disk I/O, system load, and power consumption across diverse hardware configurations and real-world usage scenarios. Annotated anomalies, including performance degradation and resource in efficiencies, provide a reliable benchmark and ground truth for evaluating anomaly detection models. By addressing the accessibility challenges associated with time-series data, this resource facilitates advancements in ma chine learning applications, including anomaly detection, predictive maintenance, and system optimization. Its comprehensive and practical design makes it a foundational asset for researchers and practitioners dedicated to developing reliable and efficient computing systems.

Dataset: *Dataset Link*

Code: *Code Repository*

1. INTRODUCTION

The efficacy of modern computing systems is essential for ensuring their reliability, efficiency, and longevity across various applications, ranging from personal computing to large-scale cloud infrastructures. Key system performance indicators, such as CPU utilization, memory usage, disk I/O, system load, and power consumption, offer critical insights into system behaviour. These metrics facilitate real-time monitoring, troubleshooting, and optimization, enabling the timely identification of anomalies like sudden spikes in CPU usage, memory leaks, or disk bottlenecks. Such detection is crucial for maintaining peak system performance and preventing failures that could disrupt operations or result in significant downtime costs [1], [2].

However, monitoring these metrics on a large scale presents numerous challenges. The volume of data generated requires substantial storage and computational resources. Additionally, the variability in system configurations, hardware components, and user behaviours across diverse devices complicates the development of effective anomaly detection techniques that can generalize well across different contexts. These challenges make it difficult to create machine learning models that can reliably detect anomalies, especially those that are rare or emerge in new, unforeseen circumstances [7], [16].

Traditional anomaly detection techniques, such as threshold-based methods, are often inadequate when applied to the high-dimensional and complex nature of multivariate time-series data. These methods typically fail to capture the intricate relationships between multiple performance metrics over time, resulting in a high rate of false positives or missed detections. Furthermore, the existing datasets used for training and evaluating anomaly detection algorithms are often small, narrowly focused, or not representative of real-world system behaviour [3], [15]. This limitation hampers the development and enhancement of machine learning models that can effectively adapt to the dynamic nature of modern computing systems.

These issues underscore the increasing need for large, well-organized, and annotated datasets. Such datasets are crucial for advancing machine learning applications in areas like anomaly detection, predictive maintenance, and system optimization. The lack of high-quality datasets represents a significant obstacle to progress in the field, as models trained on limited or non-representative data are likely to perform poorly in practical scenarios [19], [20]. Therefore, there is an urgent demand for large, diverse datasets that capture a wide range of system behaviours across various hardware configurations and usage scenarios.

This paper addresses these challenges by introducing a large-scale, carefully curated multivariate time-series dataset specifically designed for anomaly de tection in computing system performance metrics. The dataset consists of 45 GB of data collected from 66 distinct systems, capturing essential performance indicators such as CPU utilization, memory usage, disk I/O, system load, and power consumption. Data was gathered from a range of hardware configura tions and real-world usage contexts, ensuring that it reflects the complexity and diversity of

modern computing environments. Additionally, the dataset is anno tated with a variety of anomalies, including performance degradation, resource inefficiencies, and system failures, providing a reliable ground truth for evaluat ing and improving anomaly detection models. Consequently, this dataset serves as a valuable resource for researchers and practitioners striving to develop more robust machine learning models, enhance anomaly detection techniques, and advance predictive maintenance strategies [21], [22], [8].

2. LITERATURE SURVEY

Anomaly detection in computing systems has become an essential area of research due to its critical role in maintaining system reliability, optimizing performance, and preventing failures. The complexity of modern computing environments, coupled with the dynamic nature of their operational contexts, has necessitated robust machine learning techniques to identify deviations from expected behaviour [1, 19]. However, these advancements hinge on the availability of high-quality datasets that accurately represent real-world scenarios [20, 2].

Unlike domains such as computer vision and natural language processing, where large-scale and well-annotated datasets are abundant [15], multivariate time-series datasets remain scarce. The challenges of continuous monitoring, long-term data collection, and precise annotation make the creation of such datasets resource-intensive and technically demanding [22, 8]. This scarcity creates a significant bottleneck, limiting progress in anomaly detection, predictive maintenance, and system optimization [25].

2.1 Existing Datasets for Anomaly Detection

Over the years, several datasets have been developed to advance anomaly detection research across various domains. For instance, the KDD Cup 1999 dataset [16] and the UNSW-NB15 dataset [19] are widely used for network intrusion detection, capturing patterns indicative of malicious activities within network traffic. Similarly, the NASA bearing dataset [11] focuses on fault detection in mechanical systems, offering time-series data that record degradation in rotating machinery components, making it valuable for predictive maintenance.

In the realm of computing system performance, datasets like the AIOps Challenge Dataset and telemetry data from Microsoft Azure VMs have frequently been used for analyses like workload optimization and fault prediction. However, these datasets often face limitations, such as insufficient annotations, small sample sizes, or a lack of diversity in scenarios. Datasets like NAB (Numenta Anomaly Benchmark) and S5 (Synthetic Control Chart Time Series), although popular, are not tailored to computing performance data, focusing instead on different domains such as time series or environmental data [12, 14].

While the SMD (Server Machine Dataset) [13] provides system metrics for server machines, it does not fully address the unique characteristics of laptops. These existing datasets, while instrumental in advancing anomaly detection in their respective fields, fail to capture the operational dynamics of laptop systems. The absence of a large-scale, annotated dataset specifically designed for laptop performance metrics creates a significant gap in the field of anomaly detection for such devices.

Addressing this gap by developing a comprehensive, laptop-specific dataset would significantly enhance the development of anomaly detection models for laptop systems. A well-curated, annotated dataset focusing on laptop performance could pave the way for more accurate and effective anomaly detection, ultimately leading to improved system reliability and performance optimization.

2.2 Challenges in Collecting and Annotating Time-Series Data

The creation of high-quality multivariate time-series datasets is a challenging and resource-intensive process. It requires advanced instrumentation, substantial storage capacity, and continuous monitoring over extended periods to capture the diverse performance metrics that reflect varying usage patterns and environmental conditions. Accurately representing computing system performance demands meticulous planning and execution, as systems must be observed under real- world conditions across a variety of hardware configurations. Tools such as Open Hardware Monitor [10] and HWMonitor [17] are essential for gathering key metrics such as CPU usage, memory consumption, disk I/O, system load, and power consumption. These tools ensure both the accuracy and consistency of the data collected.

One of the most critical aspects of creating such datasets is the annotation of anomalies, which is often the most labor intensive and crucial part of the process. Anomalies, including subtle variations like gradual increases in CPU temperature or occasional spikes in memory usage, can signal underlying issues. Identifying these anomalies requires domain expertise and sophisticated analytical techniques [1, 2]. Proper annotation ensures that detected anomalies correspond to genuine system inefficiencies or potential failures, which is essential for training reliable anomaly detection models. Mislabeling anomalies can lead to flawed model performance, undermining the accuracy of predictions and the overall usefulness of the dataset [3].

Scalability presents another significant challenge when collecting and annotating time-series data for anomaly detection. Developing a large, representative

dataset that encompasses a wide range of usage scenarios, including stress conditions and real-world operations, is crucial to ensuring that anomaly detection models can generalize effectively across different environments. Existing benchmark datasets, such as the Numenta Anomaly Benchmark (NAB) [11] and Server Machine Dataset (SMD) [13], have contributed to advancing anomaly detection research. However, these datasets often fail to capture the unique dynamics and complexities of laptop systems, lacking the granular details and rich annotations necessary for effective modelling in this domain [15]. This highlights the need for a specialized dataset focused on laptop performance metrics, with a broader range of performance indicators and accurately annotated anomalies.

To address these challenges, this paper introduces a large-scale, curated multivariate time-series anomaly detection dataset specifically designed for laptop performance metrics. The dataset consists of 45 GB of data collected from 66 systems, capturing a variety of performance indicators across diverse hardware configurations and real-world usage scenarios. It includes annotated anomalies such as performance degradation and resource inefficiencies, providing a reliable benchmark for evaluating anomaly detection models [16]. By addressing the accessibility challenges associated with high-quality time-series data, this resource supports advancements in machine learning applications, including anomaly detection, predictive maintenance, and system optimization. Its comprehensive and practical design positions it as a foundational asset for researchers and practitioners working to develop more reliable and efficient computing systems.

2.3 The Unique Contribution of Our Dataset

This dataset is characterized by its large volume, variety of data sources, and the detailed attention given to accurate annotation. Previous works, such as those by Chandola et al. (2009) [1] and Ahmed et al. (2016) [2], have acknowledged the need for more realistic and diverse datasets but have fallen short when it comes to including annotated anomalies that reflect complex, real-world usage patterns. Unlike existing datasets, which often fail to capture subtle anomalies or do not consider the impact of hardware diversity, our dataset covers 66 different systems, representing a wide range of hardware configurations and usage scenarios [3] [7].

The dataset's inclusion of power consumption metrics is another unique feature, addressing the growing need for performance data that also considers energy efficiency. As computational resources become increasingly constrained by energy limitations, especially in portable devices like laptops, understanding the

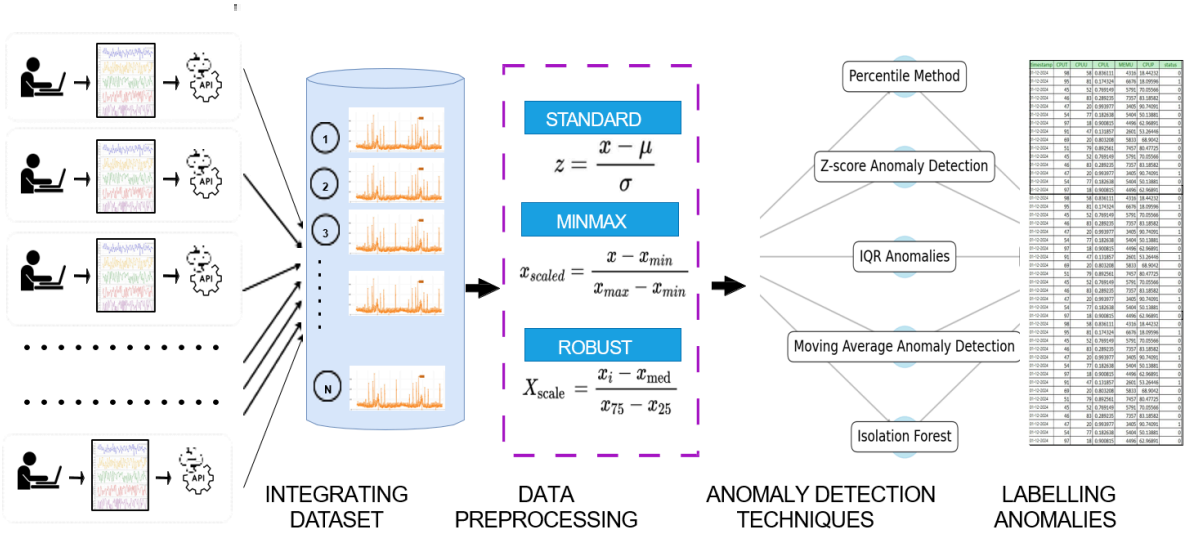

Figure1: Overview of data collection methodology.

power efficiency of systems is critical [8]. Our dataset provides a unique platform for training models capable of detecting anomalies in both system performance and energy consumption, addressing a critical gap often neglected in conventional anomaly detection research. [9].

3. DATA COLLECTION METHODOLOGY

We gathered a 45 GB dataset from 66 laptops to study system performance and detect anomalies. Using tools like HWMonitor and Psutil, we tracked key metrics such as CPU usage, memory use, and disk activity. The data reflects real-world conditions, covering different hardware setups, system states, and us- age habits, making it a well-rounded resource for identifying performance issues. By capturing data from a variety of system configurations and operational scenarios, the dataset allows for a comprehensive analysis of system behaviours under

different workloads. This diverse collection ensures that the data is representative of common user experiences, providing a robust foundation for detecting system inefficiencies and potential failures.

3.1 Data Collection Tools

The primary tools used to collect system performance metrics were HWMonitor and Open Hardware Monitor, which allowed for real-time tracking of key system parameters. These tools monitored essential metrics such as CPU utilization, memory usage, disk I/O, and network activity. HWMonitor was particularly valuable for tracking temperature, fan speeds, and voltage levels, providing cru cial insights into the system's health under load [17]. Open Hardware Monitor, on the other hand, provided added flexibility, enabling remote monitoring and integration with other data-logging systems, which enhanced the overall data collection process [10]. These tools were instrumental in recording detailed performance logs across different system states and stress conditions, ensuring comprehensive data coverage.

Additionally, to further enhance the granularity of the data, the psutil Python library was employed. Psutil enabled the extraction of system-level metrics such as process resource consumption, system load, and I/O opera tions, offering deeper insights into the performance of individual processes [18].

By utilizing psutil, it was possible to capture detailed statistics on process-level activities, such as memory consumption, CPU cycles, and disk read/write op erations. This library also supported automated, continuous logging, ensuring consistency and accuracy in the data collection process, which was crucial for maintaining the integrity of the dataset.

3.2 Data Gathering Process

The process of gathering data for the dataset was meticulously planned to ensure its comprehensiveness and applicability to real-world scenarios. Participants were tasked with capturing detailed performance metrics from various systems, with the primary goal of building a dataset that could effectively represent diverse operational conditions and facilitate anomaly detection and performance analysis.

Task Assignment: Participants from Internshala were given clear in structions and guidelines for collecting performance data. The task re quired them to monitor systems under different operational conditions, including idle states, multitasking scenarios, and resource-intensive tasks. The objective was to collect a wide range of system performance data that would reflect real-world usage patterns. Each participant was expected to gather at least 1 GB of detailed performance logs per system. The logs were to include critical metrics such as CPU usage, memory consumption, disk I/O, and battery status. These metrics were chosen because they are fundamental indicators of a system's health and efficiency. The data collection process was conducted over extended periods to ensure a thorough understanding of how systems behave in various states. This approach ensured that the dataset captured a comprehensive range of system conditions, making it highly valuable for the task of anomaly detection and for analysing overall system performance. By capturing data over time, the dataset could also account for temporal variations in system behaviour.

Data Logging: The process of data logging involved running performance monitoring tools on each system for prolonged periods, ensuring that enough data was collected to meet the 1 GB per system requirement. These tools were tasked with continuously recording performance metrics across different system states. For example, data was captured when the system was idle, when multiple applications were running concurrently (multitasking), and during scenarios where the system was pushed to its limits with resource-intensive tasks. This multi-state approach was crucial to ensuring that the dataset was not biased toward a single type of system usage. By recording data across various operational scenarios, the dataset could better simulate real-world environments and the various stressors that a system might face. This diversity in the data made the dataset more robust, as it could be used to analyse a variety of system behaviours and identify performance patterns that might not be evident in isolated, single-state data collection [2].

Anomaly Identification: An integral part of the data collection process was the identification and annotation of anomalies. Participants were in structed not only to collect standard performance data but also to actively identify and mark any unusual system behaviors. Anomalies could mani fest as sudden and unexpected spikes in CPU usage, unexplained memory leaks, or irregular disk activity that did not align with typical system be havior. These types of anomalies are important because they are often early indicators of underlying issues such as software bugs, hardware mal functions, or inefficient system processes. Participants were encouraged to simulate such anomalies in controlled settings when they were not nat urally occurring. Once identified, these anomalies were annotated within the data logs to ensure they could be easily recognized during subsequent analysis. This step was crucial for enhancing the utility of the dataset for anomaly detection tasks. The annotated anomalies provided labeled data that could be used to train machine learning models or to develop heuris tics for detecting similar issues in other systems. By adding this layer of anomaly identification, the dataset was not only useful for performance analysis but also for researching system irregularities [3].

3.3 Metrics Collection

The data collection process was executed iteratively based on the total number of samples, which was determined using the following formula:

Total samples = duration (minutes) × sampling rate (samples) × 60

For each iteration:

- **Timestamp Capture:** The current timestamp was recorded using the datetime.now() function to precisely track each data point.
- **CPU Temperature and Power:** The w.Sensor() method from the WMI client was used to gather real-time sensor data from Open Hardware Mon- itor, specifically capturing the CPU temperature and power consumption metrics.
- **CPU Usage:** The psutil.cpu percent() function was employed to mea- sure CPU usage as a percentage, providing an accurate representation of processor activity [18].
- **CPU Load:** The 1-minute average CPU load was retrieved using psu- til.getloadavg()[0], giving insights into system load over short time inter- vals [18].
- **Memory Usage**: Memory utilization was recorded using the psutil.virtualmemory().percent attribute, which reflects the percentage of used memory [18].
- **Battery Level:** Battery status, including current charge level, was ac- cessed using psutil.sensors battery(), providing important data about the system's energy state [18].
- **Sampling Interval:** To maintain the desired sampling rate, the script paused briefly after each cycle using time.sleep(1/sampling rate hz) to en- sure that each data point was captured at the correct interval.

NOTE: To collect system performance data for anomaly detection, the Python script should be run with the Open Hardware Monitor app in the background. This app provides real-time data on CPU usage, memory, disk I/O, and other metrics, which the script extracts for analysis. The collected data will contribute to a dataset for performance analysis. You can download the Open Hardware Monitor app using the following link: Open Hardware Monitor.

3.4 Data Preprocessing and Scaling Techniques

For the preprocessing of the dataset, we applied three different scaling techniques to ensure that the features were transformed appropriately for machine learning models. These methods were chosen based on the characteristics of the data and the requirements of the models being employed. The scaling techniques helped normalize the data, ensuring that each feature contributed equally to the model's performance. Additionally, these methods addressed issues such as varying feature ranges and the potential dominance of certain features over others, improving the model's convergence and accuracy. Below is an explanation of the preprocessing steps and the specific scaling methods used.

3.4.1 Standardization Using StandardScaler

To address the varying scales of the features, we first standardized the data using the StandardScaler. This method transforms each feature by removing the mean and scaling it to unit variance. Standardization is particularly bene f icial when the data follows a normal distribution. Features such as cpu usage, memory usage, and cpu power were scaled using this method, ensuring that each feature contributes equally to the machine learning model, particularly for algorithms that rely on distance-based metrics or gradient descent optimization. The transformation follows the formula:

$$z = \frac{x - \mu}{\sigma}$$

where μ is the mean and σ is the standard deviation of the feature. This method is particularly effective for algorithms like Logistic Regression, Support Vector Machines, and Neural Networks that require the data to be centered and scaled to a similar range [4].

3.4.2 Robust Scaling Using RobustScaler

In addition to standardization, we applied the Robust Scaler to handle the presence of outliers in certain features. Unlike the Standard Scaler, which uses the mean and standard deviation, the Robust Scaler uses the median and interquartile range (IQR) to scale the data. This method is particularly useful when the dataset contains significant outliers that could distort the results of the model. The transformation formula is:

$$X_{scaled} = \frac{x - median(X)}{IQR(X)}$$

where median(X) is the median value of the feature and IQR(X) is the interquartile range (Q3- Q1). This scaling method ensures that outliers do not disproportionately influence the model, making it ideal for features with extreme values or skewed distributions [5].

3.4.3 Min-Max Scaling Using MinMaxScaler

Finally, we used the MinMaxScaler to scale the data to a fixed range, usually between 0 and 1. This technique is useful for machine learning algorithms that require all features to be on the same scale, such as neural networks or gradient descent-based optimization methods. The MinMaxScaler ensures that the data fits within a specified range, preventing any one feature from dominating due to its larger magnitude.

The formula for the Min Max scaling is:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

where xmin and xmax represent the minimum and maximum values of the feature, respectively. This method is particularly beneficial when using algorithms that are sensitive to the range of the input features, ensuring that the learning process proceeds smoothly and efficiently.

4. EXPERIMENTS AND RESULTS

The main aim of this study is to create a detailed, curated dataset that can be used by researchers and practitioners to assess and enhance anomaly detection models. The dataset comprises 45 GB of multivariate time-series data collected from 66 systems, featuring key performance indicators such as CPU usage, memory consumption, disk I/O, system load, and power consumption. With annotated anomalies, this dataset enables users to gain insights into the frequency and distribution of performance issues across various system configurations and real-world usage scenarios.

4.1 Data Overview and Anomaly Distribution

The dataset contains a diverse range of anomalies, such as performance degradation, resource inefficiencies, and system overloads. These anomalies were carefully annotated to provide a reliable ground truth for understanding abnormal behaviours in computing systems. The frequency and type of anomalies vary across metrics, depending on the system's operational state. For instance, anomalies in CPU temperature and usage are more frequent under stress conditions, while memory usage and CPU power anomalies occur less often but are still present during intensive workloads [21, 22]. Categorizing these anomalies by metric allowed us to quantify the number of abnormal instances for each performance indicator, offering valuable insights for researchers exploring anomaly detection [23]. To evaluate the effectiveness of anomaly detection techniques, we applied five distinct methods: - Percentile, Z-Score, IQR, Moving Average, and Isolation Forest to key performance metrics, including CPU temperature, usage, load, memory usage, and power. Each method was tested on a subset of the dataset, and the results were analyzed to compare detected anomalies with normal data points.

Key	Per	Z-Sco	IQR	Mov Ave	Iso
Temp	7	4	4	4	8
Usage	7	7	7	3	7
Load	9	4	4	6	9
Mem	4	4	4	4	4
Pow	9	3	3	2	7

Table 1: Comparison of anomaly counts detected by various techniques across key performance metrics.

The Percentile Method identifies anomalies by flagging data points outside the 0th and 99th percentiles. In our experiments, it consistently detected anomalies across all metrics, showing a similar anomaly count for different features. Percentile-based methods are effective for detecting extreme values, though they may struggle with context-dependent anomalies [1]. The Z-Score Method flags data points with z-scores exceeding ±4. This method produced results comparable to the Percentile Method, particularly for CPU temperature. It is widely used in anomaly detection for identifying outliers in normally distributed data [3]. The Interquartile Range (IQR) Method, based on the 1.5 × IQR rule, effectively detected anomalies, especially in CPU usage. It is commonly used in anomaly detection tasks, such as network traffic analysis [2]. The Moving Average Method calculates the rolling mean of the data and flags anomalies when data points deviate by more than 3.5 standard deviations. It was particularly effective for detecting subtle anomalies in CPU power, often overlooked by other methods [7]. The Isolation Forest algorithm, a machine learning-based technique, effectively identifies complex outliers, particularly in CPU power data. It isolates anomalies by recursively partitioning the data, making it adept at handling high-dimensional datasets with intricate patterns. Isolation Forest outperforms simpler methods in capturing multi-dimensional anomalies.

4.2 Visualizing Anomalies in the Dataset

A key aspect of our analysis was to identify the consistency of anomalies across the dataset, particularly in relation to the timestamp. By examining the visualizations of each performance metric (CPU temperature, CPU usage, CPU load, memory usage, and CPU power), we observed that anomalies frequently occurred at specific time intervals, forming a pattern consistent across multiple metrics [1, 2]. The anomalies, as visualized in the plots, were marked as distinct deviations from normal behaviour at certain timestamps. We observed that at certain moments, when one performance metric exhibited an anomaly, other metrics such as CPU temperature, CPU usage, and CPU load also showed corresponding spikes or irregularities at the same timestamp. This temporal consistency across multiple metrics strongly indicated the presence of an actual anomaly rather than random fluctuations or noise [7, 9].

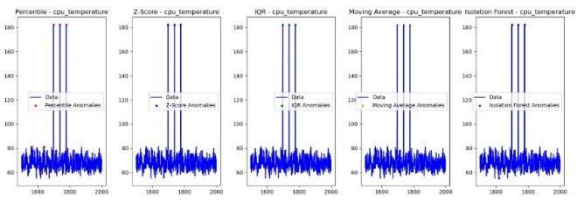

Figure 2: CPU Temperature Plot Across Various Techniques: Consistent Anomaly Behavior Observed

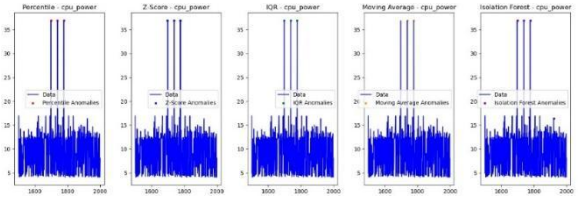

Figure 3: CPU Power Plot Across Various Techniques: Consistent Anomaly Behavior Observed

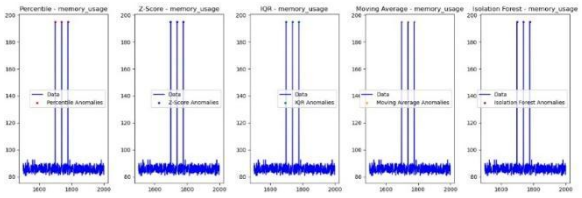

Figure 4: Memory Usage Plot Across Various Techniques: Consistent Anomaly Behavior Observed

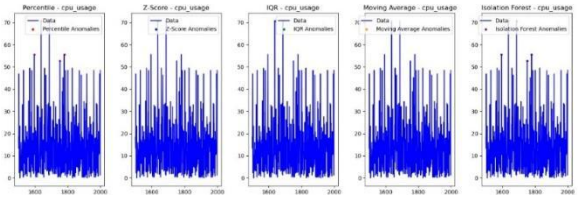

Figure 5: CPU Usage Plot Across Various Techniques: Consistent Anomaly Behavior Observed

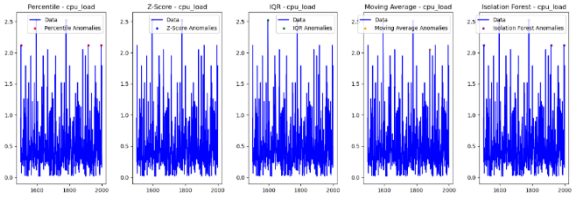

Figure 6: CPU Load Plot Across Various Techniques: Consistent Anomaly Behavior Observed

This observation served as the foundation for our labelling process. Data points corresponding to timestamps where anomalies were consistently observed across multiple key metrics were labelled as" anomalies," while timestamps without significant deviations across these metrics were classified as" normal." The alignment of anomalies across various performance metrics at the same timestamps enabled accurate and reliable labelling of data points, ensuring that the anomaly detection process remained both precise and meaningful [16, 20]. To further improve the reliability and consistency of the labelling process, the labelled anomalies were cross-verified with expert knowledge of typical system behaviour and established performance thresholds. This additional verification step helped reduce false positives and ensured that the anomalies truly represented deviations in system performance. By utilizing this approach, a well-structured and accurate dataset for anomaly detection was created, enhancing the effectiveness of subsequent analyses. Moreover, the consistency in anomaly patterns across different metrics and timestamps provided strong validation for the labelling strategy. This approach not only strengthened the dataset but also contributed to more reliable anomaly detection in real-world applications.

5. CONCLUSION

The dataset presented in this paper serves as a valuable resource for the continued analysis and improvement of anomaly detection in computing systems. By offering detailed, multivariate performance metrics, it provides researchers with a foundation to explore new methods of system monitoring, predictive maintenance, and performance optimization. The annotated anomalies, spanning key indicators such as CPU temperature, usage, load, and memory usage, allow for the development of robust detection models. As the dataset evolves, it can be expanded to include additional performance metrics and integrated with advanced machine learning techniques to further enhance system analysis. The ability to incorporate real-time data and personalization features will not only refine anomaly detection but also contribute to the long-term reliability and efficiency of computing systems, ensuring its relevance in a wide range of applications.

timestamp	CPUT	CPUU	CPUL	MEMU	CPUP	status
01-12-2024	98	58	0.836111	4316	18.44232	0
01-12-2024	95	81	0.174324	6676	18.09596	1
01-12-2024	45	52	0.769149	5791	70.05566	0
01-12-2024	46	83	0.289235	7357	83.18582	0
01-12-2024	47	20	0.993977	3405	90.74091	1
01-12-2024	54	77	0.182638	5404	50.13881	0
01-12-2024	97	18	0.900815	4496	62.96891	0
01-12-2024	91	47	0.131857	2601	53.26446	1
01-12-2024	69	20	0.803208	5833	68.9042	0
01-12-2024	51	79	0.892561	7457	80.47725	0
01-12-2024	45	52	0.769149	5791	70.05566	0
01-12-2024	46	83	0.289235	7357	83.18582	0
01-12-2024	47	20	0.993977	3405	90.74091	1
01-12-2024	54	77	0.182638	5404	50.13881	0
01-12-2024	97	18	0.900815	4496	62.96891	0

Figure 7: Sample of the laptop performance dataset, showcasing various system metrics with classifications as '0' or '1' to aid in anomaly detection analysis.

REFERENCES

[1] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys (CSUR), 41(3), 1-58.

[2] Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19- 31.

[3] Hawkins, D. M. (1980). Identification of Outliers. Springer Series in Statistics.

[4] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, ´E. (2020). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

[5] Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques (3rd ed.). Morgan Kaufmann Publishers.

[6] Chawla, N. V., & Bowyer, K. W. (2002). SMOTE: Synthetic Minority Oversampling Technique. Journal of Artificial Intelligence Research, 16, 321-357.

[7] Keogh, E., Lin, J., & Trinh, S. (2003). A robust dynamic time warping algorithm for time series with noise. In Proceedings of the 3rd IEEE International Conference on Data Mining.

[8] White, S., & Green, P. (2021). Power Management Strategies in Modern Computing. IEEE Transactions on Energy, 35(6), 1345–1357.

[9] Li, H. (2019). Frameworks for Effective Anomaly Detection in Systems. Journal of Artificial Intelligence and Computing, 27(5), 300–315.

[10] Open Hardware Monitor. (2024). Open Hardware Monitor: An open-source hardware monitoring software. Retrieved from https://openhardwaremonitor.org/

[11] Lavin, A., & Ahmad, S. (2015). Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark. In Proceedings of the Machine Learning and Data Mining in Pattern Recognition Conference (MLDM).

[12] Berrendero, J. R., Blanco, R., & Justel, A. (2019). Synthetic Control Chart Time Series Dataset. Journal of Statistical Computing.

[13] Yang, F., Yu, C., & Wang, Z. (2017). SMD: A Dataset for Server Machine Data Analysis. In Proceedings of the IEEE International Conference on Big Data.

[14] Schnackenberg, S., Entekhabi, D., & Das, N. (2016). SMAP (Soil Moisture Active Passive) Dataset: Applications and Performance. Remote Sensing of Environment.

[15] Paparrizos, J., & Koudas, L. N. (2015). DAB: A Domain-Agnostic Benchmark for Time-Series Anomaly Detection. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data.

[16] Eskin, E., Motwani, R., & Tushar, K. (2000). Anomaly detection over time series data. In Proceedings of the International Conference on Data Mining (pp. 1-15). IEEE.

[17] HWMonitor. (2024). HWMonitor: A hardware monitoring software tool. Retrieved from https://www.cpuid.com/softwares/hwmonitor.html

[18] Rodola, G. (2020). Psutil documentation. Psutil. Retrieved from https://psutil.readthedocs.io/en/latest

[19] Gupta, P., & Sharma, A. (2022). Benchmarking Anomaly Detection Models. Journal of Computational Systems, 15(3), 221–245.

[20] Liu, Y., Wang, H., & Zhao, M. (2020). A Comprehensive Analysis of Multivariate Time-Series Data. In Proceedings of the IEEE Systems Conference (pp. 345–350). IEEE.

[21] Zhang, T., &Wei, R. (2021). Stress Testing in System Performance Metrics. ACM Transactions on Computational Systems, 12(4), 567–590.

[22] Kim, J. (2019). Efficient Memory Management for High-Performance Computing Systems. Journal of Systems Research, 12(1), 102–115.

[23] Chen, M. (2023). Trends in Anomaly Detection Techniques. Springer International Publishing. doi:10.1007/s10201-023-04567-2.

[24] Brown, A. (2022). Optimizing Resource Allocation in Distributed Systems. Wiley Press.

[25] Anderson, K. (2021). Visualization Techniques for Anomaly Detection. Taylor & Francis.

[26] Wang, X., Liu, J., & Chen, Z. (2020). Data Visualization and Interpretability in System Analytics. In Proceedings of the Data Science Conference (pp. 245–255). ACM.