AI

MDPI

Article

Machine Learning-Based Network Anomaly Detection: Design, Implementation, and Evaluation

Pilar Schummer [1], Alberto del Rio [2,*], Javier Serrano [3], David Jimenez [4], Guillermo Sánchez [5] and Álvaro Llorente [2]

[1] Escuela Técnica Superior de Ingenieros de Telecomunicación (ETSIT), Universidad Politécnica de Madrid, 28040 Madrid, Spain; pilar.schummer.bengoa@alumnos.upm.es
[2] Signals, Systems and Radiocommunications Department, Escuela Técnica Superior de Ingenieros de Telecomunicación (ETSIT), Universidad Politécnica de Madrid, 28040 Madrid, Spain; alvaro.llorente@upm.es
[3] Informatic Systems Department, Escuela Técnica Superior de Ingeniería de Sistemas Informáticos (ETSISI), Universidad Politécnica de Madrid, 28031 Madrid, Spain; javier.serrano@upm.es
[4] Physical Electronics, Electrical Engineering and Applied Physics Department, Escuela Técnica Superior de Ingenieros de Telecomunicación (ETSIT), Universidad Politécnica de Madrid, 28040 Madrid, Spain; david.jimenezb@upm.es
[5] Global CTIO Unit, Telefónica Innovación Digital, 28050 Madrid, Spain; guillermo.sanchezillan@telefonica.com
* Correspondence: a.delriop@upm.es

Abstract: Background: In the last decade, numerous methods have been proposed to define and detect outliers, particularly in complex environments like networks, where anomalies significantly deviate from normal patterns. Although defining a clear standard is challenging, anomaly detection systems have become essential for network administrators to efficiently identify and resolve irregularities. **Methods:** This study develops and evaluates a machine learning-based system for network anomaly detection, focusing on point anomalies within network traffic. It employs both unsupervised and supervised learning techniques, including change point detection, clustering, and classification models, to identify anomalies. SHAP values are utilized to enhance model interpretability. **Results:** Unsupervised models effectively captured temporal patterns, while supervised models, particularly Random Forest (94.3%), demonstrated high accuracy in classifying anomalies, closely approximating the actual anomaly rate. **Conclusions:** Experimental results indicate that the system can accurately predict network anomalies in advance. Congestion and packet loss were identified as key factors in anomaly detection. This study demonstrates the potential for real-world deployment of the anomaly detection system to validate its scalability.

Keywords: anomaly detection; explainable AI; machine learning; network anomalies; network performance

Citation: Schummer, P.; del Rio, A.; Serrano, J.; Jimenez, D.; Sánchez, G.; Llorente, Á. Machine Learning-Based Network Anomaly Detection: Design, Implementation, and Evaluation. *AI* **2024**, *5*, 2967–2983. https://doi.org/10.3390/ai5040143

Academic Editors: Stephen Ojo, Agbotiname Lucky Imoize and Lateef Adesola Akinyemi

Received: 4 November 2024
Revised: 5 December 2024
Accepted: 13 December 2024
Published: 17 December 2024

1. Introduction

Data networks have revolutionized modern communications, making connectivity and speed essential in today's world. An efficient data system, which encompasses the technological infrastructure that enables communication and information exchange between devices, is crucial [1]. Ensuring correct data movements and promptly detecting and resolving problems is critical [2]. Anomaly detection systems have emerged to fulfill this need, identifying key performance indicators and understanding data characteristics [3,4]. These systems, often based on machine learning (ML), use advanced algorithms to detect anomalies, thus ensuring the integrity of the environment [5–7].

Anomaly prediction proactively prevents future errors and protects sensitive information and critical infrastructures from potential attacks [8,9]. This study aimed to address the need for effective anomaly detection systems in network infrastructures, which is crucial for IT professionals who face new threats on a daily basis.

Detecting anomalies is essential for maintaining and preserving network systems. Information technology professionals face continuous challenges due to the evolution of Internet networks and communication systems. The development of advanced anomaly detection systems is crucial to identify anomalies and respond effectively, avoiding collateral damage. Anomalies can manifest themselves in a variety of ways, such as unauthorized access, network crashes, or unusual traffic patterns. Although existing solutions have achieved high theoretical results, they often lack the categorization of network data [10–12]. This study aimed to develop an anomaly detection system that considers the network environment, traffic situations, and dataset variables, creating a prototype usable in real security systems. The project consists of the implementation of autonomous anomaly detection algorithms integrated into a data application for user interaction and deployment.

This work focused on detecting point anomalies in network traffic data by analyzing individual feature values and identifying outliers that significantly deviate from expected behavior [13]. Point anomalies, or deviations in single data points, represent a foundational approach in anomaly detection as they indicate isolated irregularities without considering the interactions between variables. For instance, a sudden spike in packet loss rate would be flagged as an anomaly based solely on this feature's deviation from the baseline. By establishing a clear methodology for identifying such patterns, this study sets the groundwork for future research into more complex anomaly types, such as collective anomalies, which assess correlations between multiple network features [14].

The main objective of this study was to design and implement artificial intelligence (AI) algorithms for network anomaly detection, analyzing network anomalies to develop a system capable of identifying anomalous patterns and behaviors. This study established a solid groundwork for identifying individual anomalies and aimed to develop sophisticated detection methods that can identify interconnected patterns within the network, offering a more holistic strategy for monitoring network health.

2. The State of the Art

Anomaly detection is a critical process for identifying unusual patterns or outliers [15] in data that significantly deviate from the norm, i.e., instances of data that stand out as being different from all others, resulting in an important problem that has been well studied in various research areas and application domains [16]. These anomalies can indicate major problems such as security breaches, system malfunctions, or fraudulent activities, making anomaly detection a vital component in various domains such as network security, finance, and healthcare. From [16], the problem has been posed as three types of approaches: unsupervised, supervised, and one-class. Unsupervised are algorithms that have to detect anomalies without prior labeling [17,18], supervised are algorithms that work with labeled datasets (although they are usually very unbalanced as the anomaly rate is usually quite low), and the one-class classification is for when there are no anomalous samples as part of the training data [19]. There are several approaches to anomaly detection, each with its strengths and applications and different computational complexity; the state-of-the-art research techniques can be divided into different categories based on the underlying assumptions and the approach taken, where the mainstream is based on deep learning-based anomaly detection.

Statistical methods are among the first approaches, relying on probabilistic models to identify data points that deviate from the expected behavior. When the generating process exhibits anomalous behavior, it results in the formation of outliers. As a result, outliers provide valuable information about unusual characteristics of systems and entities [20]. Outlier detection algorithms provide two types of results: outlier scores, which classify data points according to their degree of "outlierness"; and binary labels, which classify data points as outliers or not. While scores provide detailed information, binary labels, which are usually obtained by setting thresholds on the scores, provide a simplified and actionable result that is often used in practical applications. Thresholding techniques involve setting predefined thresholds for specific metrics, where any data point that exceeds

these thresholds is flagged as anomalous. For example, if normal CPU usage hovers around 30%, a peak of 90% would be considered an anomaly. Another common statistical approach is principal component analysis (PCA), which reduces the dimensionality of the dataset, allowing for the identification of anomalies in a transformed space in which unusual data points stand out [21].

In recent years, ML-based methods have gained prominence due to their ability to handle complex, high-dimensional data. However, these systems are not always well designed to address the key problem of working with data that are often unbalanced [22]. Supervised learning techniques, such as Support Vector Machines (SVMs) [23] and neural networks [24,25], require labeled datasets in which anomalies are explicitly identified during the training phase. These models then learn to distinguish between normal and anomalous data points based on the labels provided. However, the challenge often lies in the availability of labeled data, particularly for anomalies, which are, by nature, rare and diverse [26]. Tang et al. [27] introduced a graph neural network (GNN)-based approach for anomaly detection, highlighting how anomalies affect graph spectral properties.

To address the scarcity of labeled data, unsupervised learning methods are widely used. These techniques, such as k-means clustering and isolation forests, do not require labeled training data. Instead, they detect anomalies by identifying data points that significantly deviate from the learned structure of normal data. For example, in clustering, anomalies are identified as points that do not fit well into any of the established clusters, suggesting that they are outliers [28]. Other techniques applied are autoencoders, which do not require labeled data and instead focus on identifying deviations from the learned patterns of normal network behavior [29]. For example, they can reconstruct normal network traffic with high fidelity, and any significant deviation in reconstruction error can be flagged as an anomaly [30]. Recent advancements in anomaly detection have emphasized the importance of leveraging multi-dimensional models for explainability and robustness. Heine et al. [31] proposed the cellwise estimator, which uses OLAP cubes to model normal network behavior across various aggregation levels and detects deviations as anomalies. Aiello et al. [32] demonstrated how k-means clustering combined with rule-based learning can detect anomalies, such as DNS tunneling, in streaming data.

Hybrid methods combine the strengths of supervised and unsupervised learning. Semi-supervised learning, for example, uses a small amount of labeled data to guide the training of a model on a larger set of unlabeled data, making it especially useful when it is difficult to find labeled anomalies [33,34]. Ensemble methods further improve detection by combining multiple models, such as clustering algorithms and decision trees, to improve overall accuracy and robustness [35,36]. In this field, in addition to autoencoders, methods based on generative adversarial networks (GANs) have gained popularity for their ability to detect anomalies by generating realistic representations of normal data. A GAN consists of two neural networks: a generator G, which takes a vector z sampled from a simple prior distribution (latent space) and maps it to an artificial data space with the same dimensions as the real data; and a discriminator D, which is responsible for distinguishing between real and generated data. A prominent example is GANomaly [37], a semi-supervised method that uses a generator to create normal data, as well as a discriminator that learns to distinguish between generated data and anomalies. This approach has proven effective in detecting anomalies in images and videos by identifying cases that cannot be faithfully reproduced by the generator, indicating anomalous behavior [38–42].

In network security, where timely anomaly detection is crucial, these techniques are used to monitor and analyze traffic patterns. These tasks are achieved using methods such as clustering or ML algorithms, and network anomalies such as unauthorized access or distributed denial-of-service (DDoS) attacks can be detected based on deviations from normal traffic behavior [43,44]. ML models are also being integrated into network security architectures in innovative ways, especially in edge computing instances. This is especially beneficial in distributed network environments, such as the Internet of Things (IoT), where data must be processed and analyzed quickly to avoid security breaches [45,46]. In these

environments, the potential for vulnerabilities is very high due to the nature of edge devices, where the lack of continuous updates or outdated firmware are common problems [47]. Furthermore, as networks evolve to a cloud continuity paradigm, potential attacks can affect not only IoT sensors and actuators, but also ubiquitous computing elements. In this context, network communication now plays an important role in connecting services that were previously centralized and easily controllable.

3. Materials and Methods

This section details the methodology and materials used in the development and implementation of the network anomaly detection system. It includes the design of the network environment, the data collection process, and the ML techniques employed for anomaly detection.

3.1. Overview of the Research Design

The methodology for this study on network anomaly detection is structured around several essential phases. The process begins with selecting critical network traffic metrics and constructing a complete dataset that represents both typical and anomalous network behaviors. This dataset is generated within simulated network environments, which are designed to encompass a variety of scenarios that closely mimic real-world conditions. The next phase is data preprocessing, where the collected data are normalized, and key features are extracted. Techniques like PCA are applied to reduce dimensionality, which helps streamline ML models without losing critical information.

The core of this research focuses on the application and comparison of various ML models, including both supervised and unsupervised methods. These models are trained and tested on the preprocessed data, and cross-validation is used to evaluate their performance in detecting anomalies in the network.

This research also included an evaluation framework using metrics such as accuracy, precision, recall, and F1-score. These metrics help measure the effectiveness of each model in detecting anomalies and minimizing false positives. In addition, this study examined the integration of the best performing models into a real-time anomaly detection system using edge computing to enable rapid detection and response in networked environments.

3.2. Network Environment and Data Collection Methods

The network environment and topology design for this study are intended to simulate real-world conditions under which network anomalies may occur. The design includes a virtual network environment in which two network topologies are configured to represent typical enterprise and service provider networks. This configuration allows network scenarios to be generated and analyzed in a controlled way.

The emulated network scenario was developed using the EVE-NG version 4.0.1-86-PRO (https://www.eve-ng.net/, accessed on 21 October 2024) tool. The network topology is composed of multiple interconnected nodes, including routers and servers, which are configured to simulate different traffic patterns and loads. The topology also incorporates different types of network links with varying bandwidths and latencies.

In the initial configuration, as shown in Figure 1 (left), the network applied a 'Best Effort' routing strategy from Server 1 to both Server 2 and Server 3 using the most direct path with the fewest hops to optimize data delivery. In the second topology configuration (right), the network's routing strategy shifted from a 'Best Effort' approach to a 'Fallback Path'. This change simulates a scenario in which primary links between routers near Server 2 and Server 3 are broken, requiring alternative data routes. Removing these links forces the network to reroute traffic along secondary paths, enabling the collection of new data to assess the response and performance under these changed conditions.

A custom network probe utilizing sockets, developed in Python 3.9, was implemented on each server to gather network traffic data. This probe measured key network parameters,

including bandwidth, throughput, latency, jitter, congestion, and packet loss. The data collection process was organized into several stages.

First, the network probe was deployed on virtual machines, and it was configured to capture detailed network metrics in real time. This setup allowed for continuous monitoring and data collection. Next, to simulate realistic network traffic, multimedia content (video files) was streamed between servers using FFmpeg version 4.4.2 (https://www.ffmpeg.org/, accessed on 9 October 2024). Video streams were encoded and transmitted at different bitrates, representing diverse network load levels and providing a basis for evaluating network performance under varying traffic intensities.

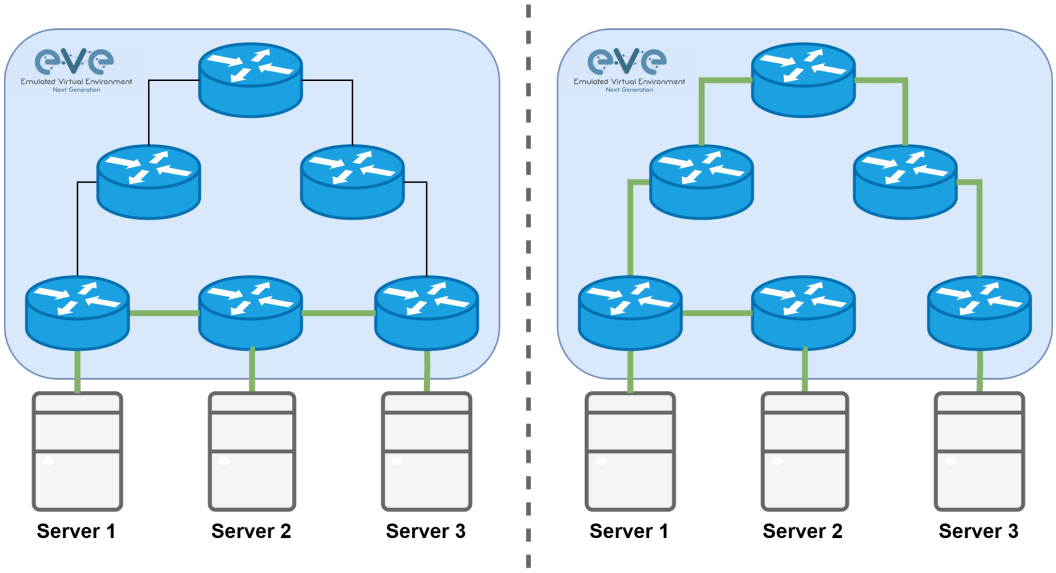

Figure 1. Network topology design: (**left**) best effort path; (**right**) fallback path.

The experimental setup involved four primary scenarios: a base scenario with typical network traffic; individual video streaming with a single video stream; simultaneous video streaming with multiple concurrent video streams; and adaptive streaming, where the video bitrate was randomly adjusted to gather new data.

All collected data were stored in a structured CSV file, recording details such as network status metrics (bandwidth, throughput, latency, jitter, congestion, and packet loss) and scenario-specific information, including the active network topology, routing path, destination server, and video configuration. This structured approach enables a clear and organized view of the network's behavior under each test condition.

The final test plan, as shown in Table 1, outlines each scenario configuration. Each test case specifies the type of scenario, the applied path (best effort or fallback), and the respective server destinations.

Table 1. Network performance metric scenarios.

ID	Scenario	Path	Source	Destination
1	Base	Best effort	Server1	Server2
2	Base	Best effort	Server1	Server3
3	Individual	Best effort	Server1	Server2
4	Individual	Best effort	Server1	Server3
5	Simultaneous	Best effort	Server1	Server2
6	Simultaneous	Best effort	Server1	Server3
7	Base	Fallback	Server1	Server3
8	Individual	Fallback	Server1	Server3
9	Simultaneous	Fallback	Server1	Server3
10	Adaptive	Fallback	Server1	Server3

4. Experimental Framework

This section outlines the experimental configuration used to assess network anomaly detection. Initially, we employed a simulated network environment mirroring real-world scenarios, and then we conducted a statistical examination of the essential network metrics. Lastly, we utilized t-Distributed Stochastic Neighbor Embedding (t–SNE) for visualizing data patterns and correlations, facilitating the development of the model.

4.1. Performance of Simulated Network Environment

The network was designed to replicate various real-world conditions, such as bandwidth limitations, latency, and packet loss, which are key indicators of network health. However, certain limitations inherent in the virtualized environment impacted network performance, particularly the fixed bandwidth of 2 Mbps, which is the maximum capacity of the emulated configuration. The developed dataset included the following key features.

In the course of the simulation, network traffic was produced in a managed environment, with data recorded every 15 s across various scenarios. This process yielded a dataset comprising 1000 entries, the performance details of which are outlined in Table 2. For instance, the throughput values averaged 1.91 Mbps, but ranged from 0.90 Mbps to 7.14 Mbps depending on the scenario. This variation highlights the network's ability to handle varying loads, although the upper range exceeds the expected bandwidth due to the simulated conditions.

Congestion was another critical metric, showing a wide range from 0.03% during minimal traffic to more than 134. 37% when the network was heavily loaded. This indicates that the network experienced severe congestion under certain conditions, which is essential to test the robustness of the anomaly detection models. An average packet loss of 5.43% was recorded, with peaks of up to 50%, suggesting that the network struggled to maintain data integrity in high volumes of traffic.

Table 2. Summary statistics of the network performance metrics.

Metric	Throughput (Mbps)	Congestion (%)	Packet Loss (%)	Latency (ms)	Jitter (ms)
Mean	1.91	23.86	5.43	54.75	0.86
Std	0.90	32.14	9.65	275.62	0.89
Min	0.05	0.03	0	4.48	0
25%	1.43	0.09	0	6	0.48
50%	1.98	10.74	0	7.55	0.65
75%	2.46	40.89	7.50	10.29	0.94
Max	7.14	134.37	52.5	3051.58	10

Latency measurements also varied significantly, from as low as 4.48 ms to extreme values such as 3051 ms, especially in scenarios with heavy traffic and congestion. Such high latency indicates severe performance degradation, which would likely affect time-sensitive applications.

Jitter, the measure of variability in packet delay, showed the least variation between scenarios, indicating that, despite the other performance issues, the temporal consistency of the packet delivery was relatively stable. This is an important consideration, particularly for applications such as video streaming, where jitter can significantly affect quality.

Furthermore, the network was tested for its ability to handle varying loads by altering the network overhead percentage, which ranged from 0% to 120%. This was achieved by streaming multimedia content at different bitrates, with the number of simultaneous video streams varying from 0 to 6. The results of these scenarios highlight the limitations of the network, particularly its bandwidth constraints and susceptibility to congestion and packet loss under high load conditions.

4.2. Statistical Analysis

An exploratory data analysis (EDA) was conducted to examine the characteristics and distributions of the collected network traffic data. This analysis focused on assessing the correlations between various network metrics to uncover underlying patterns and trends. Figure 2 presents a correlation matrix illustrating the relationships among network metrics. Throughput exhibited a moderate negative correlation with congestion (−0.42) and video occupancy percentage (−0.44), suggesting that increased congestion and video occupancy are associated with a lower throughput. Additionally, congestion shows a strong positive correlation with both packet loss (0.45) and video occupancy percentage (0.70), implying that as congestion rises, packet loss and video occupancy also increase, which can negatively impact network performance.

Furthermore, the video occupancy percentage and bitrate were perfectly correlated (1.00), as expected, since bitrate directly influences video occupancy. The number of videos also showed a moderate correlation with congestion (0.36) and packet loss (0.22), indicating that a higher number of concurrent video streams may contribute to increased congestion and packet loss.

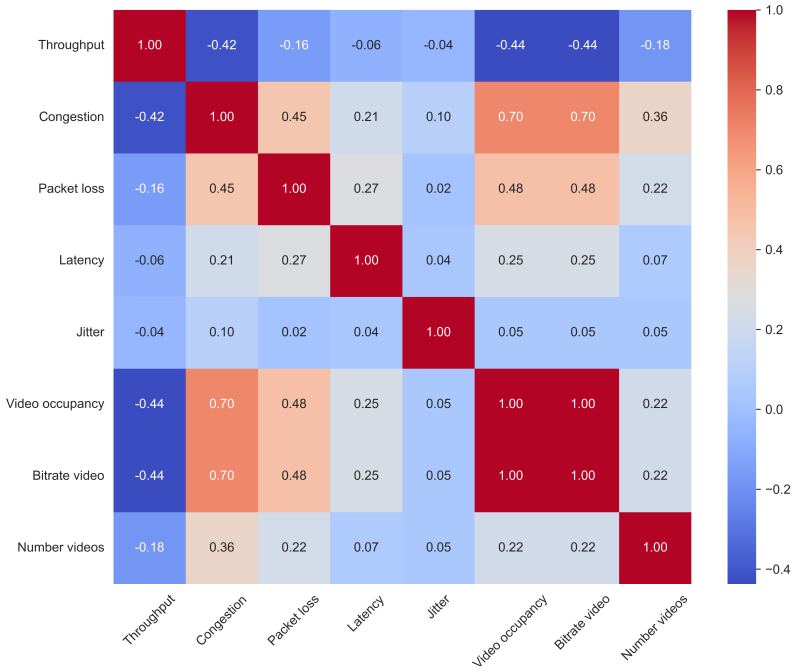

Figure 2. Correlation matrix.

In addition, Figure 3 provides a detailed analysis of the network metrics, highlighting the variations in the throughput, latency, and jitter across the 10 scenarios. The first subset illustrates the throughput distribution for each cluster, revealing that the scenarios with minimal network traffic—specifically, Base Scenarios 1, 2, and 7—achieved the highest throughput levels. This was expected as the probe utilizes maximum channel capacity when secondary traffic is minimal. In contrast, scenarios with higher secondary traffic exhibit reduced throughput due to bandwidth sharing constraints. Additionally, the throughput remained consistent across both the best–effort routes (Scenarios 1 to 6) and fallback routes (Scenarios 7 to 10), indicating that the route type has minimal impact on throughput performance

The second subset depicted latency variations. The analysis revealed that each router introduced approximately 2 milliseconds of latency. For the best–effort routes, the observed latency ranged between 6 and 7 ms, while the extended routes exhibited latency between 10 and 12 milliseconds. In addition, the data showed a latency difference of

about 2 milliseconds depending on the server generating the traffic, which aligns with the addition or removal of a router along the route.

The third subset examined the jitter behavior. The correlation matrix above indicated no significant correlation between jitter and other variables. Consistent with this observation, the jitter boxplots showed notable increases in Scenarios 4 and 8, which align with the periods of the highest average jitter observed in the overall jitter plot. These increases occured in scenarios with a higher number of hops, suggesting that additional nodes along the transmission path raise the probability of delay fluctuations, thus increasing jitter. Each additional node introduced potential delays, contributing to the observed jitter variations.

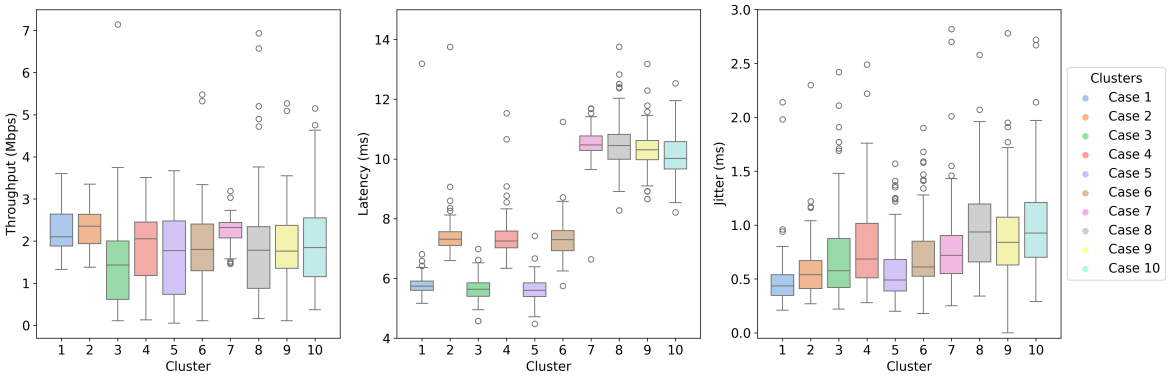

Figure 3. Combined boxplots.

Given the high number of features in the dataset, a 2D representation becomes practical only with dimensionality reduction techniques, such as t–SNE. This method enables a reduction in data dimensionality, facilitating the visualization of data distribution and the relationships between data points, which supports a deeper analysis of network performance patterns and feature interactions

Figure 4 presents the t–SNE visualizations of the dataset, showing the distribution of data points for different network metrics: throughput, congestion, packet loss, latency, jitter, and router labels. The t–SNE plot for throughput shows different clusters of data points, indicating clear groupings based on throughput values. The highest values are located in the lower left area of the plot, while the lowest throughput values are clustered in the upper area. This separation suggests that the performance significantly influences the distribution of the data, with higher throughput data forming separate clusters from lower throughput data.

The congestion plot reveals that high network congestion values are clustered in the upper area of the graph. This clustering is consistent with the patterns observed in the throughput and packet loss plots, where low throughput and high packet loss are associated with high congestion. The clear separation of congestion values suggests that network congestion is a critical factor in data distribution.

The packet loss plot shows that the data points with the highest packet loss values are located in the upper area of the graph. This pattern coincides with the throughput plot, where the areas with the lowest throughput correspond to those with the highest packet loss. In contrast, packet loss values lower than 30% are mainly found in the lower-left area, indicating a clear relationship between low packet loss and higher throughput.

The t–SNE plot for latency shows that data points are broadly distributed across the plot, with higher latency values concentrated on the far right, while lower latency values are mainly located on the left. In the middle region, both high and low latency values are mixed, indicating a more diverse distribution compared to other metrics.

The jitter plot does not show the same distinct clustering patterns observed in the previous plots. This was expected since jitter had no significant correlation with the other variables in the dataset. The distribution of high jitter values appeared more scattered,

indicating that jitter is influenced by factors other than those affecting throughput, packet loss, and congestion.

The last plot corresponds to the path of the data points according to the two main routes in the dataset: the best–effort route (blue) and the extended route or fallback (orange). As seen in the plot, the jitter values tend to be higher in areas with higher number of routers, which is consistent with the argument that additional nodes introduce more delay variability, thus increasing jitter.

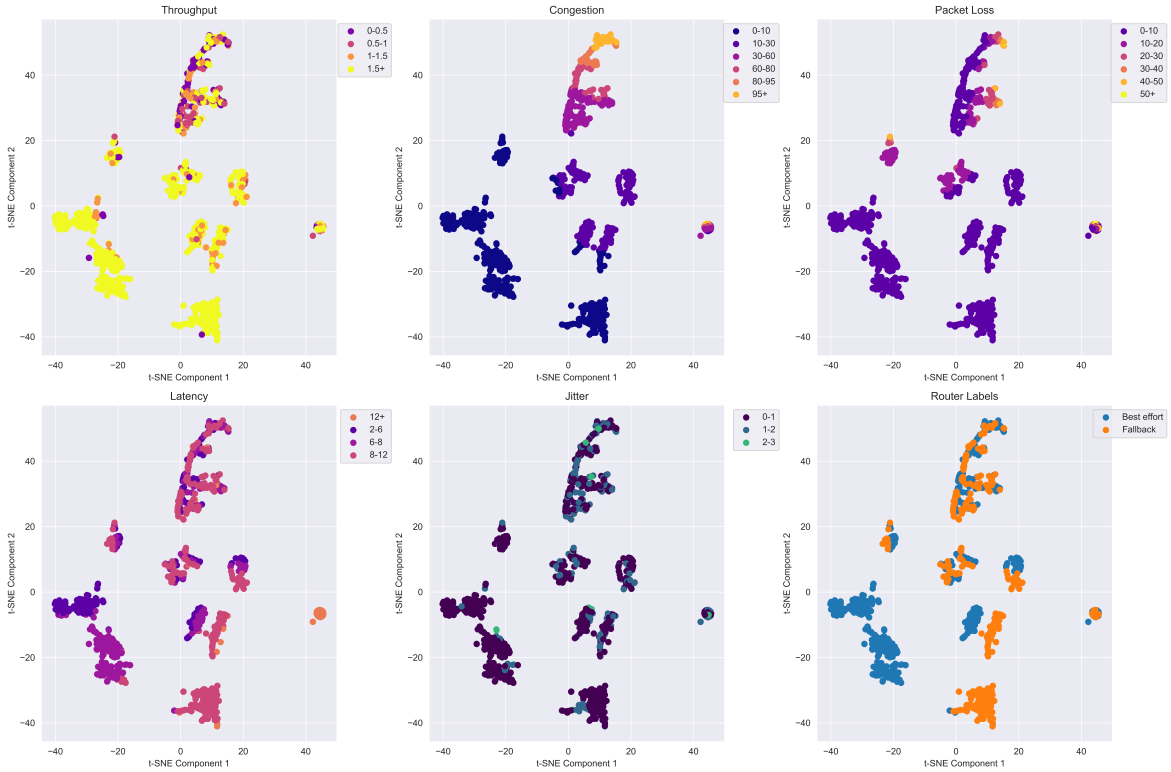

Figure 4. t–SNE visualizations for various network metrics: (**top-left**) throughput; (**top-middle**) congestion; (**top-right**) packet loss; (**bottom-left**) latency; (**bottom-middle**) jitter; and (**bottom-right**) path.

5. Results

This section presents the results from unsupervised and supervised models used to detect network anomalies. We first explored unsupervised methods to find patterns and shifts in the metrics like throughput. Next, we assessed the supervised models for classifying anomalies with labeled data. Finally, we examined model interpretability to see how each metric influences anomaly detection.

5.1. Unsupervised Learning Models

Unsupervised learning models operate without labeled data, relying instead on the ability to autonomously identify patterns and insights. This study employed two key techniques for unsupervised anomaly detection: change point detection and clustering [48]. The first technique involves detecting change points (CPs) in time series data using the Dynp tool version 1.1.9 (https://centre-borelli.github.io/ruptures-docs/user-guide/detection/dynp/, accessed on 17 October 2024). This approach applies dynamic programming to segment the time series into smaller subproblems, analyzing the statistical properties of each segment to identify significant changes in the data. These CPs represent temporal anomalies, where abrupt changes in network metrics may indicate underlying problems. For example, Figure 5 on the left illustrates the time series of the throughput metric in various test plan scenarios. The detected change points, marked with dashed red lines,

occurred precisely when the throughput dropped to lower values, specifically between 0 and 2 Mbps. This drop was reflected in the green cluster, which indicates periods where the potential errors or performance problems were likely to have occurred.

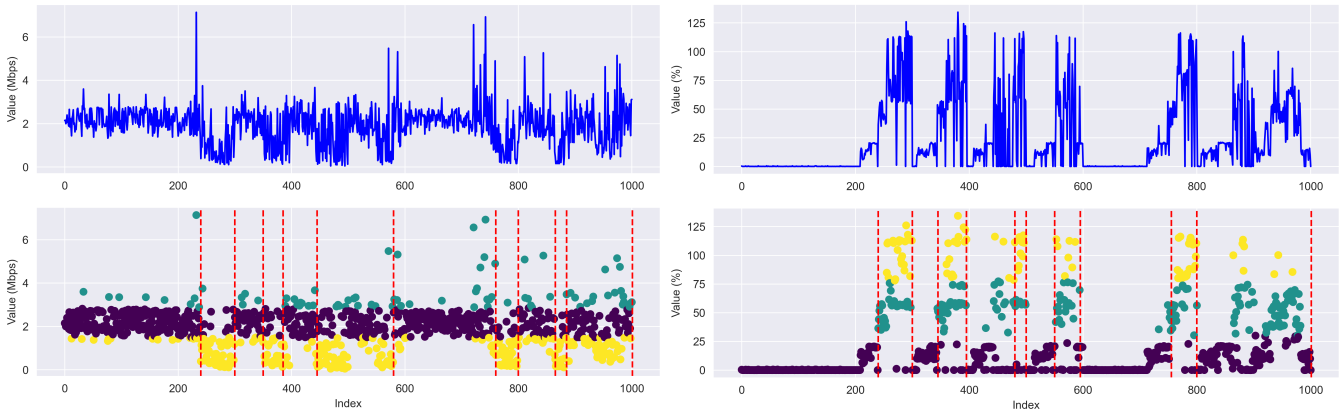

Figure 5. Comparison of time series: (**left**) throughput; (**right**) congestion.

To complement the change point detection, a clustering approach, centered on k-means clustering, was employed. While k-means was chosen for its simplicity and interpretability, the distance metric used was dynamic time warping (DTW) [49] rather than the traditional Euclidean distance. DTW is particularly effective in aligning sequences of different lengths and capturing nonlinear temporal distortions, which is essential in the accurate clustering of time series data. As shown in Figure 5 on the right, this approach is applied to the congestion metric. Here, change points are detected when the congestion exceeds 60%, which corresponds to the yellow cluster. The analysis reveals that, despite some isolated values of high congestion toward the end of the series, they do not form CPs due to their isolated nature, unlike the consistent pattern observed in the earlier part of the series.

Another application of these methods is seen in the analysis of the packet loss metric, as shown in Figure 6 on the left. The clustering method identifies significant anomalies when the packet loss exceeds 20%, and it is represented by the green dots in the figure. These anomalies are classified as severe, while the yellow cluster captures minor anomalies, providing a nuanced understanding of the network behavior under varying conditions.

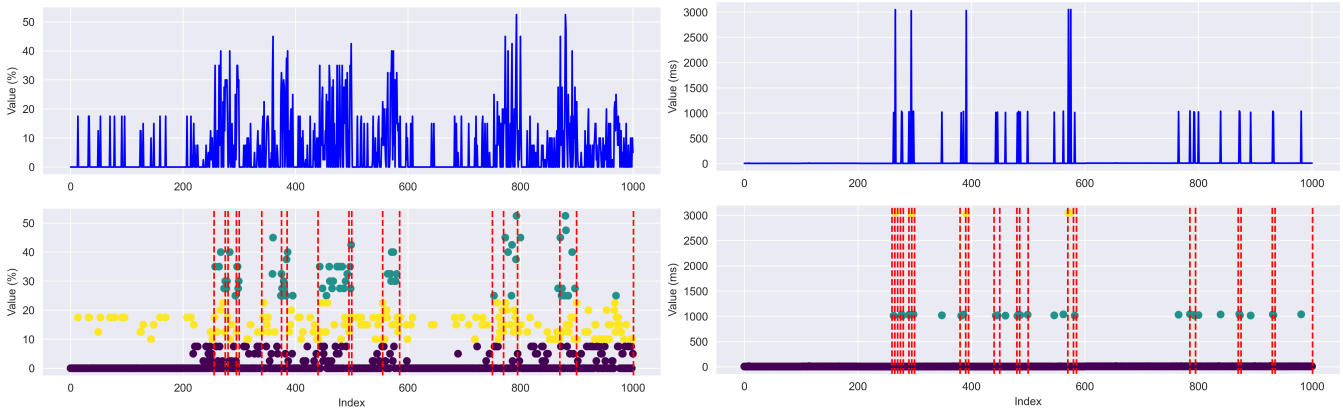

Figure 6. Comparison of time series: (**left**) packet loss; (**right**) latency.

Finally, analysis of the latency metric, as shown in Figure 6 on the right, did not yield clear conclusions. Although there are noticeable spikes in latency, reaching up to 3000 ms, the change point detection and clustering do not reveal a consistent pattern of anomalies. The detected change points and clustering are scattered, making it difficult to identify specific causes or patterns. This suggests that latency alone may not provide sufficient

information, and that further analysis with additional metrics may be necessary to fully understand these latency spikes.

5.2. Supervised Learning Models

An anomaly is typically defined as one or more observations that significantly deviate from the rest of the data. However, there is no universally accepted definition among scientists, leading to variability in how anomalies are identified, especially in corporate environments. Each organization typically specifies its own thresholds and methods for anomaly detection based on its unique challenges and requirements.

Given the diversity of methods available to define anomalies, a common approach is the use of statistical methods. A simple and widely recognized technique for identifying anomalies in one-dimensional data following a normal distribution is to use the mean and standard deviation. This method labels points as anomalies if they deviate from the mean by more than three standard deviations. However, this technique assumes that the data follow a normal distribution, which is not applicable to the dataset used in this study.

To address the need for a more flexible and robust approach, especially considering the non-normal distribution of the data, non-parametric techniques are preferred. Non-parametric methods make minimal assumptions about the underlying distribution of the data, which makes them suitable for a wide range of datasets. In this study, the Tukey method [50] was used to detect global outliers. This method defines outliers based on the interquartile range (IQR) and is formulated as follows:

$$\text{Outlier} = \begin{cases} \text{True}, & \text{if } x_i < Q1 - k \cdot \text{IQR or } x_i > Q3 + k \cdot \text{IQR} \\ \text{False}, & \text{otherwise}, \end{cases} \quad (1)$$

where:

- x_i is the value of the data point.
- $Q1$ is the first quartile (25th percentile).
- $Q3$ is the third quartile (75th percentile).
- IQR is the interquartile range ($Q3 - Q1$).
- k is a threshold, typically set to 1.5 for standard outlier detection.

Data points that fall below $Q1 - 1.5 \cdot \text{IQR}$ or above $Q3 + 1.5 \cdot \text{IQR}$ were labeled as anomalies. This approach effectively captures outliers without assuming a specific distribution, making it suitable for the diverse characteristics of the dataset in this study. Finally, the evolution of the anomalies for throughput, congestion, packet loss, and latency is depicted in Figure 7.

Once the anomalies were labeled, some ML models were evaluated. The models included Logistic Regression, Random Forest, and SVM. All of them offered different metrics to compare with each other, such as accuracy, precision, recall, and F1-score, which are summarized for all models in Table 3.

First, the Logistic Regression model, a simple binary classifier, was applied. Despite its simplicity, the model achieved an accuracy of 91.2%, effectively classifying a significant portion of the data. However, it showed limitations in capturing the complex non-linear relationships present in the dataset, resulting in a notable number of misclassifications, especially in distinguishing between true anomalies and normal variations in the data.

To address these challenges, more sophisticated models, such as Random Forest and SVM, were also implemented. The Random Forest model, known for its robustness and ability to handle a large number of input features, improved accuracy to 94.3%. This model's ensemble approach, combining multiple decision trees, allowed it to better capture the complexities of network traffic data, reducing the incidence of false positives and negatives.

The SVM model further improved detection accuracy, achieving a rate of 96.5%. By identifying the optimal hyperplane that separates normal data points from anomalies, the SVM provided more nuanced classification, especially in cases where the decision boundary between classes was not linear. Despite its higher computational cost, SVM was effective in

handling the complexity of the dataset, although it slightly overestimated the number of anomalies with a false positive of 6%.

Nevertheless, higher accuracy or precision does not always mean a better model. The actual anomalies in the dataset based on the labeling process were 8.29%, and only Random Forest was able to approach this value. However, SVM demonstrated the highest accuracy in classifying anomalies within the dataset, and Logistic Regression also showed a tendency to slightly overestimate the number of anomalies, resulting in a higher false positive rate.

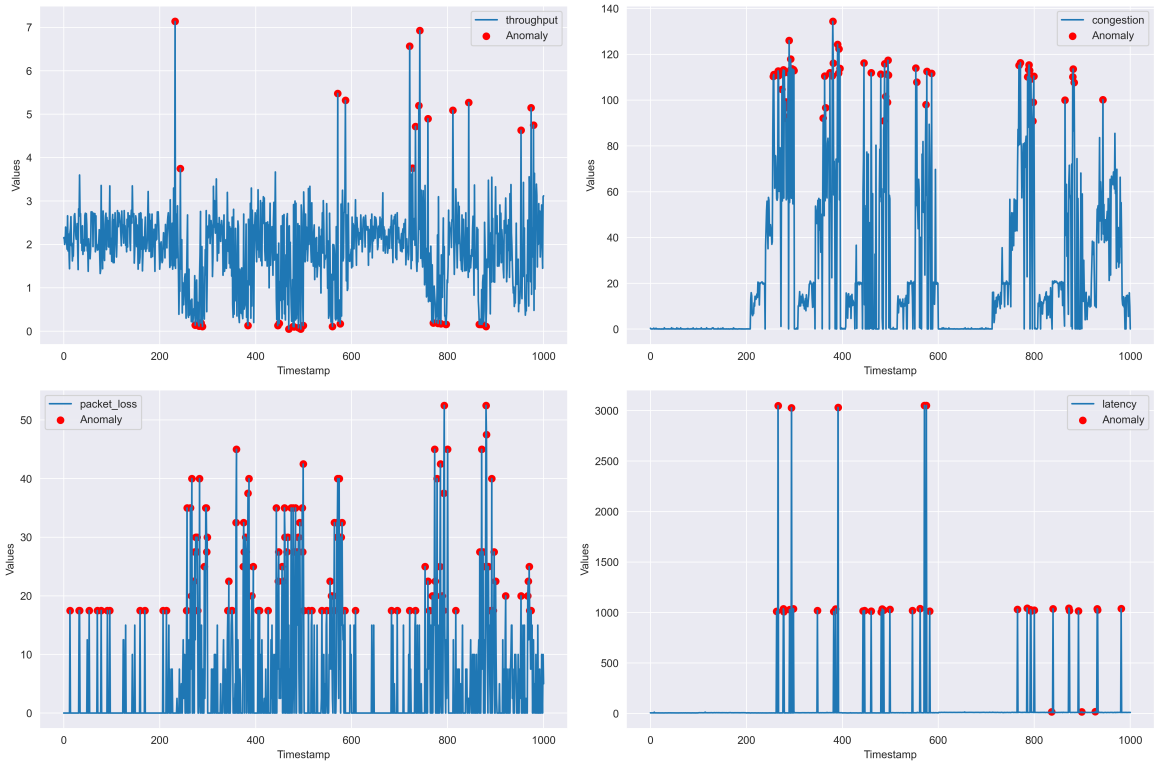

Figure 7. Anomalies defined based on the Tukey method.

Table 3. Classification reports for Logistic Regression, Random Forest, and Support Vector Machine.

Metric	Logistic Regression	Random Forest	Support Vector Machine
Accuracy	91.2%	94.3%	96.5%
Precision	89.7%	93.8%	95.9%
Recall	90.5%	94.0%	96.2%
F1-score	90.1%	93.9%	96.0%
Anomalies	15.9%	8.9%	11.9%

5.3. Model Explainability

In addition to the accuracy metrics, model interpretability was improved through the use of explainability methods. SHapley Additive exPlanations (SHAP) were used to identify the features that most influenced model predictions, showing how different network metrics, such as congestion and packet loss, influenced anomaly detection.

Figure 8 presents the explainability results, showing that the most influential feature is congestion, where higher values have a substantial impact on the prediction, with SHAP values close to 0.5. This indicates that congestion is a critical factor in the model decision making process. The next most significant metrics are latency and packet loss. In addition, low throughput rates influence predictions, highlighting the importance of throughput in identifying anomalies.

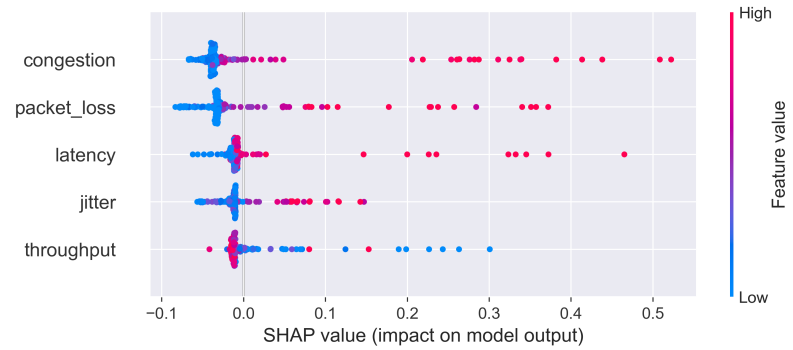

Figure 8. SHAP values impact metrics.

As shown in Figure 9 (non-anomalous instance), we observed how different features influenced the model prediction for a particular data point where no anomaly was detected. The base value (0.95) represents the average model result over the entire dataset, indicating a high probability that the data are normal. In this case, the most significant factor pushing the prediction toward a higher value (non-anomalous) is the low throughput value (0.83 Mbps), which slightly decreased the prediction but was outweighed by the absence of packet loss and the relatively moderate level of congestion (20.55%). The low latency (9.76 ms) also supported the model decision to classify this instance as non-anomalous. The overall prediction (f(x) = 0.48) remained close to the threshold but indicated that this instance is more likely to be normal due to the balanced contributions of the features.

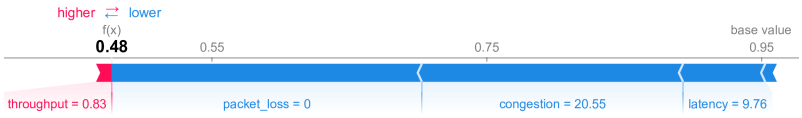

Figure 9. SHAP example with non anomaly data.

The second observation included in Figure 10 illustrates the feature contributions for a data point that the model has identified as anomalous. Here, the base value starts at 0.01, reflecting the average probability that a point is anomalous in the dataset. The features significantly contribute to push the prediction toward identifying this case as an anomaly (f(x) = 0.48).

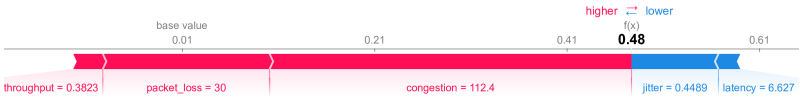

Figure 10. SHAP example with anomaly data.

The throughput value of 0.3823 Mbps and the high packet loss rate of 30% strongly influenced the model to classify this instance as anomalous, as indicated by the large red bars that increase the prediction. Additionally, the high congestion value (112.4%) further strengthened the likelihood that this was an anomaly. On the other hand, the relatively low jitter (0.4489 ms) and latency (6.627 ms) slightly counteracted the prediction, but their influence was not sufficient to prevent the model from classifying this case as anomalous.

6. Conclusions

This study successfully developed and assessed a ML-based system for detecting network anomalies. A realistic network environment was simulated to generate a customized dataset capturing both normal and anomalous behaviors. The analysis included both supervised and unsupervised models, with performance evaluations for each approach. The scalability of the proposed anomaly detection system was evaluated using a modular architecture, enabling deployment in distributed environments. To test integration with ex-

isting network security frameworks, we leveraged edge computing for anomaly detection in real time, ensuring compatibility with microservices-based architectures.

Unsupervised techniques, such as change point detection and clustering, effectively identified temporal anomalies in the network data. Additionally, supervised algorithms like Random Forest and SVM demonstrated high accuracy in anomaly detection, with Random Forest closely matching the actual anomaly rate in the dataset. While some models exhibited a slightly higher false positive rate, the system overall proved effective in reliably detecting network anomalies with high accuracy.

The developed anomaly detection system could be integrated into existing network security frameworks to provide real-time monitoring and detection of unusual network behavior [51,52]. This capability can identify potential security threats, such as intrusions, DDoS attacks, or data breaches as they occur, enabling immediate response and mitigation [53,54]. In addition, it can benefit from advanced networking solutions that ensure the isolation and security of virtualized workloads through microservices platforms [55].

Extending the dataset to include other network environments could have several important implications, particularly given the limitations of the network environment used in this study. Factors such as diverse protocols, varying hardware, and regional traffic characteristics were not included, and this may have not fully represented the heterogeneity of real-world networks. Including them would enhance the generalizability of the models [56,57], making them more effective in different types of networks with varying traffic patterns, protocols, and performance characteristics, especially in multi-domain environments [58]. For example, common anomalies in IoT networks, such as unusual device behavior or security breaches due to weak authentication [59], could be better detected with a more diverse dataset.

Future work could also explore detecting collective anomalies [60,61], where patterns of interdependent anomalies across multiple metrics provide deeper insights into network behavior. Integrating collective anomaly detection could uncover complex, multi-metric disruptions that may go undetected when monitoring individual features alone [62].

Potential system enhancements could include the integration of adaptive learning mechanisms that allow the models to be continuously updated and improved as new data becomes available [63,64]. Implementing a more robust feature selection process could also help in refining the model's focus on the most critical network metrics [65]. Finally, extending the explainability features to provide more detailed insights into model decisions could further help network administrators understand and address anomalies more effectively [66,67].

Author Contributions: Conceptualization, A.d.R. and J.S.; Funding acquisition, D.J.; Investigation, A.d.R.; Methodology, P.S., A.d.R. and G.S.; Project administration, J.S.; Resources, G.S.; Software, P.S.; Validation, P.S.; Writing—Original Draft, P.S.; Writing—Review and Editing, A.d.R., J.S., D.J., G.S. and Á.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially funded by the following projects: Horizon Europe CODECO project, grant number 101092696; Horizon Europe NEMO project, grant number 101070118; and Horizon Europe CyberNEMO project, grant number 101168182.

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable.

Data Availability Statement: The original data presented in the study are openly available on Kaggle at 10.34740/kaggle/dsv/9325531 [68].

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
CP	Change Point
DDoS	Distributed Denial of Service
DTW	Dynamic Time Warping
EDA	Exploratory Data Analysis
GAN	Generative Adversarial Network
IoT	Internet of Things
IQR	Interquartile Range
ML	Machine Learning
PCA	Principal Component Analysis
SHAP	SHapley Additive exPlanations
SVM	Support Vector Machine
t–SNE	t-Distributed Stochastic Neighbor Embedding

References

1. Niyato, D.; Dong, Q.; Wang, P.; Hossain, E. Optimizations of power consumption and supply in the smart grid: Analysis of the impact of data communication reliability. *IEEE Trans. Smart Grid* **2013**, *4*, 21–35. [CrossRef]
2. Lalou, M.; Tahraoui, M.A.; Kheddouci, H. The critical node detection problem in networks: A survey. *Comput. Sci. Rev.* **2018**, *28*, 92–117. [CrossRef]
3. Fernandes, G.; Rodrigues, J.J.P.C.; Carvalho, L.F.; Al-Muhtadi, J.F.; Proença, M.L. A comprehensive survey on network anomaly detection. *Telecommun. Syst.* **2019**, *70*, 447–489. [CrossRef]
4. Zhao, N.; Zhu, J.; Wang, Y.; Ma, M.; Zhang, W.; Liu, D.; Zhang, M.; Pei, D. Automatic and generic periodicity adaptation for KPI anomaly detection. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 1170–1183. [CrossRef]
5. Abdelkhalek, M.; Ravikumar, G.; Govindarasu, M. ML-based anomaly detection system for DER communication in smart grid. In Proceedings of the 2022 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), New Orleans, LA, USA, 24–28 April 2022; pp. 1–5. [CrossRef]
6. Gadal, S.; Mokhtar, R.; Abdelhaq, M.; Alsaqour, R.; Ali, E.S.; Saeed, R. Machine learning-based anomaly detection using K-mean array and sequential minimal optimization. *Electronics* **2022**, *11*, 2158. [CrossRef]
7. Zehra, S.; Faseeha, U.; Syed, H.J.; Samad, F.; Ibrahim, A.O.; Abulfaraj, A.W.; Nagmeldin, W. Machine learning-based anomaly detection in NFV: A comprehensive survey. *Sensors* **2023**, *23*, 5340. [CrossRef]
8. De Benedetti, M.; Leonardi, F.; Messina, F.; Santoro, C.; Vasilakos, A. Anomaly detection and predictive maintenance for photovoltaic systems. *Neurocomputing* **2018**, *310*, 59–68. [CrossRef]
9. Carrasco, J.; López, D.; Aguilera-Martos, I.; García-Gil, D.; Markova, I.; García-Barzana, M.; Arias-Rodil, M.; Luengo, J.; Herrera, F. Anomaly detection in predictive maintenance: A new evaluation framework for temporal unsupervised anomaly detection algorithms. *Neurocomputing* **2021**, *462*, 440–452. [CrossRef]
10. Díaz-Verdejo, J.E.; Estepa Alonso, R.; Estepa Alonso, A.; Madinabeitia, G. A critical review of the techniques used for anomaly detection of HTTP-based attacks: Taxonomy, limitations and open challenges. *Comput. Secur.* **2023**, *124*, 102997. [CrossRef]
11. Ahmed, M.; Naser Mahmood, A.; Hu, J. A survey of network anomaly detection techniques. *J. Netw. Comput. Appl.* **2016**, *60*, 19–31. [CrossRef]
12. Ajila, S.A.; Lung, C.H.; Das, A. Analysis of error-based machine learning algorithms in network anomaly detection and categorization. *Ann. Telecommun.* **2022**, *77*, 359–370. [CrossRef]
13. Russo, S.; Besmer, M.D.; Blumensaat, F.; Bouffard, D.; Disch, A.; Hammes, F.; Hess, A.; Lürig, M.; Matthews, B.; Minaudo, C.; et al. The value of human data annotation for machine learning based anomaly detection in environmental systems. *Water Res.* **2021**, *206*, 117695. [CrossRef] [PubMed]
14. Lu, T.; Wang, L.; Zhao, X. Review of anomaly detection algorithms for data streams. *Appl. Sci.* **2023**, *13*, 6353. [CrossRef]
15. Aggarwal, C.C. An introduction to outlier analysis. In *Outlier Analysis*; Springer International Publishing: Cham, Switzerland, 2017; pp. 1–34. [CrossRef]
16. Chalapathy, R.; Chawla, S. Deep learning for anomaly detection: A survey. *arXiv* **2019**, arXiv:1901.03407
17. Campello, R.J.G.B.; Moulavi, D.; Zimek, A.; Sander, J. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Trans. Knowl. Discov. Data* **2015**, *10*, 1–51. [CrossRef]
18. Kiran, B.R.; Thomas, D.M.; Parakkal, R. An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *J. Imaging* **2018**, *4*, 36. [CrossRef]
19. Wang, X.; Jin, B.; Du, Y.; Cui, P.; Tan, Y.; Yang, Y. One-class graph neural networks for anomaly detection in attributed networks. *Neural Comput. Appl.* **2021**, *33*, 12073–12085. [CrossRef]
20. Hawkins, D.M. *Identification of Outliers*; Springer: Dordrecht, The Netherlands, 1980; Volume 11. [CrossRef]

21. Shyu, M.L.; Chen, S.C.; Sarinnapakorn, K.; Chang, L. A novel anomaly detection scheme based on principal component classifier. In Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, Melbourne, FL, USA, 19–22 November 2003; IEEE Press: Piscataway, NJ, USA, 2003; pp. 172–179.
22. Gherbi, E.; Hanczar, B.; Janodet, J.C.; Klaudel, W. An encoding adversarial network for anomaly detection. In Proceedings of the Eleventh Asian Conference on Machine Learning, Nagoya, Japan, 17–19 November 2019; Lee, W.S., Suzuki, T., Eds.; PMLR: Nagoya, Japan; Volume 101, pp. 188–203.
23. Cortes, C. Support-Vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
24. Wang, S.; Balarezo, J.F.; Kandeepan, S.; Al-Hourani, A.; Chavez, K.G.; Rubinstein, B. Machine learning in network anomaly detection: A survey. *IEEE Access* **2021**, *9*, 152379–152396. [CrossRef]
25. Pang, G.; Shen, C.; Cao, L.; Hengel, A.V.D. Deep learning for anomaly detection: A review. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–38. [CrossRef]
26. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv. (CSUR)* **2009**, *41*, 1–58. [CrossRef]
27. Tang, J.; Li, J.; Gao, Z.; Li, J. Rethinking Graph Neural Networks for Anomaly Detection. *arXiv* **2022**, arXiv:2205.15508.
28. Breunig, M.M.; Kriegel, H.P.; Ng, R.T.; Sander, J. LOF: Identifying density-based local outliers. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, TX, USA, 16–18 May 2000; pp. 93–104. [CrossRef]
29. Chen, J.; Sathe, S.; Aggarwal, C.; Turaga, D. Outlier detection with autoencoder ensembles. In Proceedings of the 2017 SIAM International Conference on Data Mining (SDM), Houston, TX, USA, 27–29 April 2017; pp. 90–98. [CrossRef]
30. Javaid, A.; Niyaz, Q.; Sun, W.; Alam, M. A deep learning approach for network intrusion detection system. In Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), New York, NY, USA, 3–5 December 2016; pp. 21–26. [CrossRef]
31. Heine, F.; Kleiner, C.; Klostermeyer, P.; Ahlers, V.; Laue, T.; Wellermann, N. Detecting Attacks in Network Traffic Using Normality Models: The Cellwise Estimator. In *Foundations and Practice of Security*; Aïmeur, E., Laurent, M., Yaich, R., Dupont, B., Garcia-Alfaro, J., Eds.; Springer: Cham, Switzerland, 2022; pp. 265–282. [CrossRef]
32. Aiello, M.; Mongelli, M.; Muselli, M.; Verda, D. Unsupervised learning and rule extraction for Domain Name Server tunneling detection. *Internet Technol. Lett.* **2019**, *2*, e85. [CrossRef]
33. Shon, T.; Moon, J. A hybrid machine learning approach to network anomaly detection. *Inf. Sci.* **2007**, *177*, 3799–3821. [CrossRef]
34. Song, H.; Jiang, Z.; Men, A.; Yang, B. A hybrid semi-supervised anomaly detection model for high-dimensional data. *Comput. Intell. Neurosci.* **2017**, *2017*, 8501683. [CrossRef]
35. Pu, G.; Wang, L.; Shen, J.; Dong, F. A hybrid unsupervised clustering-based anomaly detection method. *Tsinghua Sci. Technol.* **2020**, *26*, 146–153. [CrossRef]
36. Ghrib, Z.; Jaziri, R.; Romdhane, R. Hybrid approach for anomaly detection in time series data. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–7. [CrossRef]
37. Akcay, S.; Atapour-Abarghouei, A.; Breckon, T.P. GANomaly: Semi-supervised anomaly detection via adversarial training. In Proceedings of the Computer Vision—ACCV 2018, Perth, Australia, 2–6 December 2018; Jawahar, C.V., Li, H., Mori, G., Schindler, K., Eds.; Springer: Cham, Switzerland, 2019; pp. 622–637. [CrossRef]
38. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. *arXiv* **2014**, arXiv:1406.2661. [CrossRef]
39. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. *arXiv* **2015**, arXiv:1412.6572.
40. Creswell, A.; White, T.; Dumoulin, V.; Arulkumaran, K.; Sengupta, B.; Bharath, A.A. Generative adversarial networks: An overview. *IEEE Signal Process. Mag.* **2018**, *35*, 53–65. [CrossRef]
41. Schlegl, T.; Seeböck, P.; Waldstein, S.M.; Schmidt-Erfurth, U.; Langs, G. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *Information Processing in Medical Imaging*; Niethammer, M., Styner, M., Aylward, S., Zhu, H., Oguz, I., Yap, P.T., Shen, D., Eds.; Springer: Cham, Switzerland, 2017; pp. 146–157. [CrossRef]
42. Zenati, H.; Foo, C.S.; Lecouat, B.; Manek, G.; Chandrasekhar, V.R. Efficient GAN-based anomaly detection. *arXiv* **2019**, arXiv:1802.06222.
43. Purwanto, Y.; Rahardjo, B. Traffic anomaly detection in DDos flooding attack. In Proceedings of the 2014 8th International Conference on Telecommunication Systems Services and Applications (TSSA), Kuta Bali, Indonesia, 23–24 October 2014; pp. 1–6. [CrossRef]
44. El Sayed, M.S.; Le-Khac, N.A.; Azer, M.A.; Jurcut, A.D. A flow-based anomaly detection approach with feature selection method against ddos attacks in sdns. *IEEE Trans. Cogn. Commun. Netw.* **2022**, *8*, 1862–1880. [CrossRef]
45. Patrikar, D.R.; Parate, M.R. Anomaly detection using edge computing in video surveillance system. *Int. J. Multimed. Inf. Retr.* **2022**, *11*, 85–110. [CrossRef] [PubMed]
46. Yu, X.; Yang, X.; Tan, Q.; Shan, C.; Lv, Z. An edge computing based anomaly detection method in IoT industrial sustainability. *Appl. Soft Comput.* **2022**, *128*, 109486. [CrossRef]
47. Karie, N.M.; Sahri, N.M.; Yang, W.; Valli, C.; Kebande, V.R. A review of security standards and frameworks for IoT-based smart environments. *IEEE Access* **2021**, *9*, 121975–121995. [CrossRef]
48. Skaperas, S.; Mamatas, L.; Tsaoussidis, V. A link-quality anomaly detection framework for software-defined wireless mesh networks. *IEEE Trans. Mach. Learn. Commun. Netw.* **2024**, *2*, 495–510. [CrossRef]
49. Javed, A.; Lee, B.S.; Rizzo, D.M. A benchmark study on time series clustering. *Mach. Learn. Appl.* **2020**, *1*, 100001. [CrossRef]

50. Tukey, J.W. *Exploratory Data Analysis*; Reading/Addison-Wesley: Boston, MA, USA, 1977.
51. Kathareios, G.; Anghel, A.; Mate, A.; Clauberg, R.; Gusat, M. Catch it if you can: Real-time network anomaly detection with low false alarm rates. In Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017; pp. 924–929. [CrossRef]
52. Zhao, S.; Chandrashekar, M.; Lee, Y.; Medhi, D. Real-time network anomaly detection system using machine learning. In Proceedings of the 2015 11th International Conference on the Design of Reliable Communication Networks (DRCN), Kansas City, MO, USA, 24–27 March 2015; pp. 267–270. [CrossRef]
53. Javaheri, D.; Gorgin, S.; Lee, J.A.; Masdari, M. Fuzzy logic-based DDoS attacks and network traffic anomaly detection methods: Classification, overview, and future perspectives. *Inf. Sci.* **2023**, *626*, 315–338. [CrossRef]
54. Kasim, Ö. An efficient and robust deep learning based network anomaly detection against distributed denial of service attacks. *Comput. Netw.* **2020**, *180*, 107390. [CrossRef]
55. Gonzalez, L.F.; Vidal, I.; Valera, F.; Martin, R.; Artalejo, D. A link-layer virtual networking solution for cloud-native network function virtualisation ecosystems: L2S-M. *Future Internet* **2023**, *15*, 274. [CrossRef]
56. Zhou, S.; Huang, X.; Liu, N.; Zhou, H.; Chung, F.L.; Huang, L.K. Improving generalizability of graph anomaly detection models via data augmentation. *IEEE Trans. Knowl. Data Eng.* **2023**, *35*, 12721–12735. [CrossRef]
57. Monshizadeh, M.; Khatri, V.; Gamdou, M.; Kantola, R.; Yan, Z. Improving data generalization with variational autoencoders for network traffic anomaly detection. *IEEE Access* **2021**, *9*, 56893–56907. [CrossRef]
58. Nogales, B.; Vidal, I.; Valera, F.; Sanchez-Aguero, V.; Lopez, D.R. Software-driven connectivity orchestration for multidomain network functions virtualization ecosystems. *IEEE Softw.* **2024**, *41*, 88–97. [CrossRef]
59. El-hajj, M.; Fadlallah, A.; Chamoun, M.; Serhrouchni, A. A survey of Internet of Things (IoT) authentication schemes. *Sensors* **2019**, *19*, 1141. [CrossRef]
60. Zheng, Y.; Zhang, H.; Yu, Y. Detecting collective anomalies from multiple spatio-temporal datasets across different domains. In Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, WA, USA, 3–6 November 2015; SIGSPATIAL '15; Association for Computing Machinery: New York, NY, USA, 2015. [CrossRef]
61. Bontemps, L.; Cao, V.L.; McDermott, J.; Le-Khac, N.A. Collective anomaly detection based on long short-term memory recurrent neural networks. In *Future Data and Security Engineering*; Dang, T.K., Wagner, R., Küng, J., Thoai, N., Takizawa, M., Neuhold, E., Eds.; Springer: Cham, Switzerland, 2016; pp. 141–152. [CrossRef]
62. Wang, C.; Zhou, H.; Hao, Z.; Hu, S.; Li, J.; Zhang, X.; Jiang, B.; Chen, X. Network traffic analysis over clustering-based collective anomaly detection. *Comput. Netw.* **2022**, *205*, 108760. [CrossRef]
63. Yan, X.; Xu, Y.; Xing, X.; Cui, B.; Guo, Z.; Guo, T. Trustworthy network anomaly detection based on an adaptive learning rate and momentum in IIoT. *IEEE Trans. Ind. Inform.* **2020**, *16*, 6182–6192. [CrossRef]
64. Dong, L.; Liu, S.; Zhang, H. A method of anomaly detection and fault diagnosis with online adaptive learning under small training samples. *Pattern Recognit.* **2017**, *64*, 374–385. [CrossRef]
65. Nakashima, M.; Sim, A.; Kim, Y.; Kim, J.; Kim, J. Automated feature selection for anomaly detection in network traffic data. *ACM Trans. Manage. Inf. Syst.* **2021**, *12*, 1–28. [CrossRef]
66. Chawla, A.; Jacob, P.; Farrell, P.; Aumayr, E.; Fallon, S. Towards interpretable anomaly detection: Unsupervised deep neural network approach using feedback loop. In Proceedings of the NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 25–29 April 2022; pp. 1–9. [CrossRef]
67. Brown, A.; Tuor, A.; Hutchinson, B.; Nichols, N. Recurrent neural network attention mechanisms for interpretable system log anomaly detection. In Proceedings of the First Workshop on Machine Learning for Computing Systems, Tempe, AZ, USA, 12 June 2018; MLCS'18; ACM: New York, NY, USA, 2018. [CrossRef]
68. del Rio, A.; Schummer, P. *Network-Anomaly-Dataset*; Kaggle: San Francisco, CA, USA, 2024. [CrossRef]