Received 1 February 2025; revised 1 March 2025; accepted 2 March 2025. Date of publication 17 March 2025; date of current version 31 March 2025.

Digital Object Identifier 10.1109/OJCOMS.2025.3550010

A Hybrid Model for BGP Anomaly Detection Using Median Absolute Deviation and Machine Learning

MARIA ANDREA ROMO-CHAVERO [1], GUSTAVO DE LOS RÍOS ALATORRE [1], JOSE ANTONIO CANTORAL-CEBALLOS [1] (Member, IEEE), JESÚS ARTURO PÉREZ-DÍAZ [1], AND CARLOS MARTINEZ-CAGNAZZO [2] (Member, IEEE)

[1] Tecnologico de Monterrey, School of Engineering and Sciences, 64849 Monterrey, Mexico
[2] Technology Management, LACNIC, Montevideo 11400, Uruguay

CORRESPONDING AUTHORS: J. A. CANTORAL-CEBALLOS and J. A. Pérez-Díaz (e-mail: joseantonio.cantoral@tec.mx; jesus.arturo.perez@tec.mx)

This work was supported in part by the Fondo Regional para la Innovación Digital en América Latina y el Caribe (FRIDA).

ABSTRACT Detecting anomalies in the Border Gateway Protocol (BGP) has proved relevant in the cybersecurity field due to the protocol's critical role in the Internet's infrastructure. BGP vulnerabilities can lead to major disruptions and connectivity failures, highlighting the need for early detection to maintain stable and secure Internet services. To address this challenge, our article presents an enhanced version of our previously published Median Absolute Deviation (MAD) anomaly detection system. We introduce a novel dynamic threshold mechanism that significantly enhances anomaly detection performance in BGP, achieving superior accuracy and F1-score. Through a comparative analysis of machine learning (ML) classification models—including Random Forest, Extra Trees, XGBoost, LightGBM, and CatBoost—we demonstrate that integrating our MAD detection system with these ML models can improve anomaly detection significantly. Additionally, we explore how MAD performs when combined with neural networks such as RNN, GRU, and LSTM, providing a valuable comparison between machine learning and neural network-based approaches. We evaluate the models performance in well-known events, such as CodeRed 1 v2, Slammer, Nimda, the Moscow blackout, and the Telekom Malaysia (TMnet) misconfiguration. Our results show an overall accuracy of 0.99 and F1-score of 0.98, demonstrating the effective integration of MAD and ML models for the identification of security threats. Our approach enables proactive detection with minimal computational costs and reduced preprocessing, proving that efficient anomaly detection is achievable.

INDEX TERMS Anomaly detection, border gateway protocol, machine learning, median absolute deviation, statistics.

I. INTRODUCTION

THE INTERNET is organized around numerous Autonomous Systems (AS), each being a large collection of routers (peers) managed by a single organization. These Autonomous Systems (ASes) use the Border Gateway Protocol (BGP) to share Network Layer Reachability Information (NLRI), ensuring efficient and reliable routing across the Internet [1]. To do so, BGP relies on the Transmission Control Protocol (TCP) using different types of messages: open, keepalive, update, and notification [2]. Since its creation in the late 1980s, it was assumed that the information provided by an AS was accurate, thus BGP did not provide explicit information authenticity verification [3].

Exploited BGP vulnerabilities, misconfigurations, or router/equipment failures can cause widespread disruptions, leading to cascading effects across the Internet due to BGP's distance-vector routing protocol design. When a router or autonomous system fails—whether from an attack or operational issues—BGP messages spread throughout the network, amplifying the impact.

These incidents can be compared to a "blast radius," where disruptions are mainly concentrated near the source and gradually decrease as they extend outward within the network topology (rather than geographical distance). For systems not directly connected to the origin, identifying the root cause becomes challenging, often appearing only as “anomalies” or unexpected increases in message activity.

The implications of BGP anomalies extend far beyond connectivity issues. Economically, such events can lead to substantial losses for businesses reliant on internet services, affecting e-commerce platforms, financial systems, and online services, while damaging customer trust. More critically, essential services like healthcare, emergency response, and government operations depend heavily on reliable internet connections. BGP problems can disrupt these services, causing delays in emergency response, interrupted communication, and compromised data integrity. For instance, the 2022 Rogers network outage in Canada, attributed to system deficiencies and human error, left millions without access to essential services, including mobile networks, Internet, and 911 emergency services [4]. Another major challenge is data security. Redirected or intercepted traffic can expose sensitive information, making it vulnerable to unauthorized access, modification, or deletion. This exposure can have long-lasting effects on individuals and organizations, including identity theft, and financial loss.

Considering the critical importance of detecting and mitigating BGP anomalies, it is essential to explore and develop advanced agile methods that improve detection capabilities, for example improving the detection of zero-day attacks. Recent advances in machine learning offer promising ways to enhance network anomaly detection [2], [5], [6], [7], [8], [9], [10]. This is supported given that machine learning models can analyze large amounts of network data, identify complex patterns, and learn from historical data to predict potential threats. This research aims to demonstrate that integrating machine learning models with our improved Median Absolute Deviation (MAD) detection system [11] increases detection accuracy, while avoiding complexity in preprocessing and model execution. Additionally, we include a comparison with neural network models, such as RNN, GRU, and LSTM, to evaluate MAD’s performance across different approaches and provide a broader perspective on its effectiveness. The main motivation of this research is to develop a proactive detection system capable of mitigating the impact of anomalies. Therefore, we aim to create a simple yet effective model that can detect anomalies proactively while being suitable for future real-time applications.

To achieve this, it is important to recognize and to understand different types of anomalies for developing a comprehensive BGP anomaly detection system. We examine three types of anomalies based on Al-Musawi et al.’s classification [12]: indirect, direct unintended, and link failure anomalies. Indirect anomalies are caused by external threats, such as worm attacks, that indirectly affect BGP and lead to instability. The events considered in our research include the CodeRed I v2, Slammer, and Nimda worms. Direct unintended anomalies, often due to misconfigurations, have a direct impact on BGP despite being accidental; an example analyzed in this research is the Telekom Malaysia (TMnet) misconfiguration, which caused global routing instability. Link failure anomalies result from disruptions in connections between autonomous systems, such as power outages or cable breaks, with the Moscow blackout being a key event in our study. By evaluating a variety of anomalies, we aim to develop a more generalized model capable of detecting multiple types of anomalies.

This research makes several key contributions to the field of BGP anomaly detection. First, we introduce an enhanced version of our previously proposed Median Absolute Deviation (MAD) detection system [11]. This enhancement includes a dynamic threshold that automatically adjusts based on F1-score and precision values, improving the system’s adaptability. Second, we developed a hybrid model for BGP anomaly detection by integrating our enhanced MAD detection system into machine learning models. This approach combines the strengths of statistical methods and machine learning to achieve more accurate anomaly detection. By including various types of anomalies, i.e., indirect (worm attacks), direct unintended (misconfigurations), and link failures, we adopt a diverse approach to anomaly detection, ensuring the robustness and generalizability of our model. Finally, we conduct a comparative analysis to evaluate the impact of MAD-labeled data on the accuracy of various models, including machine learning techniques such as Random Forest, XGBoost, Extra Trees, LightGBM, and CatBoost, as well as neural networks like RNN, GRU, and LSTM. Each model is tested twice: first using data labeled with the MAD detection system and then using data labeled based on the occurrence of these anomalous events. This dual evaluation allows us to assess the performance of MAD-labeled data across both machine learning and neural network models, providing a comprehensive understanding of its effectiveness. By identifying the model with the highest accuracy, our research supports efforts to find a reliable and effective approach for detecting BGP anomalies, thereby enhancing network security and stability. In summary, this study makes the following contributions:

- *Enhanced MAD Detection System:* Introducing a dynamic threshold that adjusts automatically based on F1-score and precision values.
- *Creation of a Hybrid Model for BGP anomaly detection:* Combining machine learning models with the MAD detection system to improve anomaly detection accuracy, while including various types of anomalies to ensure robustness and generalizability.
- *Comparative Analysis of Machine Learning Models:* Identifying the most accurate model for detecting BGP anomalies to support reliable detection systems and enhance network security.

The paper is structured as follows: Section II reviews related work, highlighting recent studies and advancements in the field. Section III details the MAD detection system and its enhancements, along with the proposed methodology, including dataset processing, data collection, and feature extraction. Section IV covers the machine learning models used and their hyperparameters. Section V presents the research results, including a performance evaluation and a comparative analysis of different machine learning models. Section VI discusses the findings in relation to existing approaches and discusses limitations and areas for improvement. Finally, Section VII concludes the paper with suggestions for future research.

II. RELATED WORK

The field of Border Gateway Protocol (BGP) anomaly detection has seen significant advancements in recent years. Recent research has focused on using various machine learning techniques to improve the robustness and effectiveness of detection systems [2], [5], [6], [7], [8], [9], [10]. These approaches have demonstrated the ability to analyze large amounts of routing data and identify complex patterns, making them a promising solution for cybersecurity. Different techniques have been explored, including supervised, unsupervised, and hybrid methods, each with its own strengths and weaknesses. This related work section provides a comprehensive overview of the latest advancements in this field, highlighting proposed methods, their advantages, disadvantages, and performance evaluation.

A. SUPERVISED LEARNING APPROACHES

Supervised learning techniques have been widely employed for BGP anomaly detection, using labeled data to train models for anomaly classification. For instance, Sanchez et al. [6] evaluated and compared various supervised algorithms, including Naive Bayes (NBC), Decision Trees (DT), Random Forests (RF), Support Vector Machines (SVM), and Multi-Layer Perceptron (MLP), for detecting BGP anomalies using graph features derived from AS-level topology. These algorithms were trained and tested on datasets extracted from RIS (Routing Information System) containing well-known BGP events. The SVM model demonstrated the highest AUC (0.94), being the best classifier. Despite these promising results, the computational expense of the graph features limits their real-time applicability.

Moreover, Li et al. [2] investigated the use of Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models, for anomaly detection on Slammer worm, WannaCry ransomware, and the Moscow Blackout. Their study included datasets from RIPE and Route Views, and a comparison evaluation between both datasets was made. The best results for LSTM and GRU were an overall accuracy of 0.86 and 0.88, respectively, from the Route Views dataset. However, the variability in performance across different datasets suggests the need for further optimization and evaluation of the model and on more diverse datasets.

Finally, Latif et al. [7] explored Graph Neural Networks (GNNs) for detecting BGP anomalies by comparing their performance with Multi-Layer Perceptron (MLP) and Support Vector Machine (SVM) models. The GNN model achieved an accuracy of 0.79 on a weakly supervised dataset of 300 anomalies, outperforming the mentioned models. However, the use of a weakly supervised dataset may have impacted the generalizability of the results, indicating a need for more robust training data. Additionally, GNNs can be computationally complex and may lack interpretability, posing challenges for real-world deployment.

B. UNSUPERVISED LEARNING APPROACHES

Unsupervised learning techniques offer the advantage of detecting anomalies without the need for labeled data, which can be beneficial for exploratory data analysis and identifying new types of anomalies. McGlynn et al. [8] introduced a deep learning method using auto-encoders to detect anomalous BGP updates for MOAS conflicts (Multiple-Origin AS) and prefix hijacking attacks. The experimental results achieved an F1-score of 0.82 for small-scale attacks and 0.75 for larger datasets, showcasing a decrease in performance with increasing data volume. While auto-encoders offer the advantage of unsupervised anomaly detection, their performance may be limited by the complexity and volume of routing data, indicating a need for further research to improve scalability.

Park et al. [10] analyzed machine learning models, including Autoencoder (AE), Random Forest (RF), One-Class SVM, and CNN-LSTM, combining supervised and unsupervised learning for detecting anomalies in BGP data. They used a tokenizer to preprocess the data and applied SMOTE to address data imbalance. The AE model outperformed the others, achieving an F1-score of 0.99 and AUC of 0.96. However, the reliance on specific preprocessing techniques like tokenization may limit the model's generalizability, potentially affecting its performance on diverse or evolving datasets and restricting its ability to detect only one type of anomaly, as demonstrated in this research.

C. HYBRID APPROACHES

Recognizing the complementary strengths of different detection methods, researchers have explored hybrid approaches that combine diverse techniques to address complex problems in anomaly detection.

For instance, Copstein and Zincir-Heywood [5] compared and analyzed three techniques for temporal representation of BGP data using timestamps, a sliding window buffer of sequential data, and a tap-delay-line approach. They conducted their experiments using Naive Bayes (NBC) and Decision Tree classifiers (DT), achieving the best results with an accuracy of 0.84 and a recall of 0.85. However, the approach's dependency on specific temporal representations may limit its adaptability to different network conditions.

Peng et al. [9] proposed the MAD-MulW framework, a hybrid anomaly detection model that combines LSTM-based autoencoders for processing time-series data with W-GAT modules for adaptive sample weighting. The LSTM autoencoder component helps the model learn and reconstruct patterns over time, while the W-GAT module refines detection by mitigating noise in the data. This model was evaluated with different anomaly events, such as Code Red II,

TABLE 1. Related work in BGP anomaly detection.

Author	Data	IA Technique(s)	DI	DU	IN	LF	Hybrid	Reported Performance
Copstein R. et al. [5]	BGP Data	DT, NBC	✓	-	-	-	✓	0.84 accuracy, 0.85 recall
Sanchez et al. [6]	RIPE	DT, RF, NBC, MLP, SVM	✓	✓	-	-	-	0.94 AUC
Li et al. [2]	RIPE, Route Views	LSTM, GRU	-	-	✓	✓	-	0.88 accuracy
Latif et al. [7]	BGP data	GNN	✓	✓	-	✓	-	0.79 accuracy
McGlynn et al. [8]	BGP Data	Auto-encoder	✓	✓	-	-	-	0.82 F1-score (small-scale)
Peng et al. [9]	RIPE, Route Views	LSTM-based auto-encoder	-	✓	✓	✓	✓	0.96 accuracy
Park, H. et al. [10]	BGP Data	AE, RF, One-SVM, CNN-LSTM	✓	-	-	-	-	0.96 AUC, 0.99 F1-score
Ours	RIPE	MAD & RF, ET, XGB, LGBM, CB	-	✓	✓	✓	✓	0.99 accuracy, 1.00 AUC

Note: DI = Direct Intended, DU = Direct Unintended, IN = Indirect, LF = Link Failure

Nimda, Slammer, Moscow Blackout, and Malaysian Telecom Leak, achieving an average accuracy of 0.96. The method improves efficiency and stability but requires optimization of window sizes for varied datasets and handling of computational complexity due to sample expansion.

In Table 1, we present a summary of the articles previously discussed, covering studies from 2019 to 2023. The most frequently used technique for detecting anomalies is the supervised technique, while more recent efforts have focused on unsupervised learning and hybrid models [8], [9], [10].

Despite significant advancements in BGP anomaly detection, existing methods often face limitations in computational efficiency, real-time applicability, and generalizability across different types of anomalies. As previously mentioned, many supervised learning approaches, such as those by Copstein and Zincir-Heywood [5] and Sanchez et al. [6], rely heavily on labeled data, which can limit their ability to detect zero-day attacks and reduce their adaptability to new threat patterns. Unsupervised methods, like the auto-encoder approach by McGlynn et al. [8], are effective at detecting new patterns without needing labeled data, but they often lack interpretability. This lack of clarity makes it difficult to understand and explain the detected anomalies, which is crucial for continuous mitigation efforts and for ensuring that the detection methods can be trusted and effectively applied in real-world environments. Without clear insights into how anomalies are detected, refining detection strategies and addressing root causes becomes challenging [13], which can weaken the long-term effectiveness of these models. Additionally, these unsupervised methods can struggle with scalability due to the complexity and volume of routing data.

Hybrid models, such as the one proposed by Park et al. [10], attempt to combine different methods to create more robust and effective solutions. However, the effectiveness and complexity of these approaches depend greatly on the specific techniques used. Complementing a hybrid model with supervised and unsupervised learning can enhance performance, but it often introduces challenges like increased computational complexity, overfitting, and difficulties in interpreting the combined model's results [14]. These difficulties can complicate deployment and reduce adaptability in dynamic environments and real-time applications, where quick and reliable decision-making is crucial. Furthermore, many studies focus on specific types of anomalies, which reduces their effectiveness in real-world scenarios where diverse anomalies occur.

In contrast, our approach integrates statistical methods with machine learning to enhance performance while maintaining computational efficiency. The enhanced Median Absolute Deviation (MAD) detection system dynamically adapts to traffic variability, reducing false positives and maintaining sensitivity. By relying on data variability rather than pre-learned patterns, MAD has the potential to detect unseen anomalies or zero-day attacks, offering the benefits of unsupervised learning with the added advantage of interpretability. Machine learning complements MAD by capturing complex patterns across additional features, improving the system's ability to detect anomalies that statistical methods alone might miss.

The model also has the potential to retrain itself, allowing for continuous improvement in detection capabilities. This hybrid framework combines the interpretability and low computational demands of statistical methods with the adaptability and accuracy of machine learning, enabling it to generalize across diverse BGP events effectively. Although the model has not yet been tested in real-world environments, its simplicity and low computational requirements make it a scalable and practical alternative to more complex supervised, unsupervised, and hybrid models, addressing key limitations of existing approaches.

III. MATERIALS AND METHODS

This section describes the methodology used for identifying anomalies in the Border Gateway Protocol (BGP). We detail the data collection, data preprocessing and feature extraction, and provide an overview of our previously published statistical detection system [11], detailing the enhancements made to improve its effectiveness. Finally, we conclude with the data processing and labeling for ML implementation.

A. DATA COLLECTION

The BGP data used in this study was obtained from the Réseaux IP Européens Network Coordination Centre (RIPE NCC) [15], which offers public Internet routing data through the Routing Information Service (RIS). For this research, data was extracted from two Remote Route Collectors (RRC): RRC00 in Amsterdam, Netherlands, and RRC04 at the CERN Internet Exchange Point (CIXP) in Geneva, Switzerland. These collectors were selected for their

TABLE 2. Anomaly events and extraction periods for anomalous data.

Anomaly Event	Extraction Dates (UTC)
Code Red 1v2	2001-07-19 13:20:00 - 2001-07-20 00:00:00
Slammer	2003-01-25 05:31:00 - 2003-01-25 19:59:00
Nimda	2001-09-18 13:19:00 - 2001-09-19 10:59:00
Moscow Blackout	2005-05-25 04:40:00 - 2005-05-25 10:33:00
TMnet	2015-06-12 08:43:00 - 2015-06-12 11:53:00

TABLE 3. Anomaly events and extraction periods for normal data.

Anomaly Event	Extraction Dates (UTC)
Code Red 1v2	2001-07-01 19:09:00 - 2001-07-02 05:42:00
Slammer	2003-01-05 04:30:00 - 2003-01-05 18:57:00
Nimda	2001-08-25 19:09:00 - 2001-08-26 16:42:00
Moscow Blackout	2005-05-27 13:06:00 - 2005-05-27 18:54:00
TMnet	2015-06-12 20:51:00 - 2015-06-12 23:54:00

complementary data scopes: RRC00, as a multihop collector, captures a global perspective by aggregating data from peers worldwide, while RRC04 provides detailed insights into the densely interconnected European region.

To facilitate the extraction, we utilized a Python script. The extraction was performed using the Python API *PyBGPStream* [16], and the data extracted consisted only of UPDATE messages.

1) DATES OF EXTRACTION

The extraction dates correspond to each anomalous event analyzed in this research: CodeRed 1 v2, Slammer and Nimda Internet worms, the Moscow blackout, and the TMnet misconfiguration.

Anomalous data were extracted at the specific times when these events occurred and had the most significant impact, as reported in the literature, with all sources detailed in Table 3 of our previous work [11]. For normal data, we extracted data from an average of 12 days before or after each occurrence, assuming these periods to be representative of normal conditions and ensuring dataset balance by extracting the same volume of data.

For comprehensive details on the extraction dates for both anomalous and normal data, please refer to Table 2 for anomalous data, and Table 3 for normal data. These dates apply for both route collectors RRC00 and RRC04.

B. DATA PREPROCESSING AND FEATURE EXTRACTION

Before feature extraction, several pre-processing steps were applied to prepare the data effectively while preserving its integrity and minimizing distortions that could impact detection accuracy:

- *TIMESTAMP Conversion:* The UNIX-format TIMESTAMP was converted to a readable Y-m-d H:M:S format.
- *Resampling into 3-Minute Intervals:* Data was grouped into 3-minute intervals, with each batch assigned the timestamp of its first record.
- *Handling Missing Data:* Missing values were replaced using cubic spline interpolation.

Once pre-processing was complete, feature extraction was performed to identify and utilize the most relevant metrics for anomaly detection. This approach was guided by the study of Arai et al. [17], which employed Correlation Feature Selection (CFS) to identify the most relevant features for anomaly detection. Recognizing that an increased number of features often leads to a higher computational cost [18], we selected the most consistently repeated features from Arai's experiments, together with those we found significant. This approach allowed us to maintain a total of eight features, ensuring low computational cost and feasible proactive detection. The features chosen from this research are shown in Table 4.

TABLE 4. Features extracted.

Feature	Definition	Category
1	Number of announcements	Volume
2	Number of withdrawals	Volume
3	Average AS-PATH length	AS-Path
4	Maximum AS-PATH length	AS-Path
5	Standard Deviation AS-PATH length	AS-Path
6	Number of duplicate withdrawals	Volume
7	Number of unique withdrawn prefixes	Volume
8	Record Counts	Volume

All features were calculated within 3-minute intervals to keep the data manageable and consistent. For each batch, we calculated the mean, maximum, and standard deviation of the AS Path Length. We converted the message types from categorical values to numeric, where 'A' represented Announcements and 'W' represented Withdrawals, and calculated the counts of each within every batch. Finally, we focused on withdrawal messages to identify unique withdrawn prefixes and duplicated withdrawals.

C. MAD DETECTION SYSTEM

In our previous work [11], we developed a detection system that incorporates the Median Absolute Deviation (MAD), a statistical method that measures how individual data points deviate from the overall median. This method is calculated by finding the median of the absolute differences between each data point and the overall median [19]. MAD was chosen for its ability to handle noisy and variable data effectively. Unlike mean-based methods, MAD calculates deviations using the median, making it resistant to outliers. This property ensures that the detection system remains robust, as it is not easily influenced by sudden spikes or extreme values in the data. Such robustness is particularly valuable in network environments, where noise and variability are common.

The MAD detection system works by monitoring unusual increases in the volume of consecutive BGP announcements and withdrawals within 3-minute intervals. It detects anomalies when the difference between the volume of announcements and withdrawals and their respective medians exceeds their predefined thresholds, and this condition persists for more than t minutes. Separate medians and thresholds are calculated for announcements and withdrawals to account for their distinct behaviors and distributions. Announcements and withdrawals were chosen for this study due to their strong correlation with anomalies, as revealed through our analysis. Simultaneous spikes in these metrics often reflect network instability, indicating frequent and irregular changes in routing paths. This behavior is commonly observed during events such as worm attacks, which generate abnormal traffic volumes that disrupt routing, network misconfigurations, which can propagate incorrect or unstable routes, and link failures, where sudden disconnections cause routes to be withdrawn and re-announced as the network attempts to recover. By focusing on these two features, the detection method effectively identifies patterns of instability while keeping computational costs low. This efficient approach ensures scalability and reliability, making it practical for detecting various types of anomalies in larger networks and real-time scenarios.

The system identifies anomalies using the following general condition:

$$|X_i - M| > n \times \text{MAD} \quad \text{and} \quad \text{duration}(X_i) \geq t \qquad (1)$$

where, X_i represents each data point, which corresponds to the volume of either announcements or withdrawals within a 3-minute batch, M is the median of the respective dataset X, and MAD is the median absolute deviation of that dataset. A data point X_i is classified as an anomaly only if the deviation from the respective median M, for both announcements and withdrawals, exceeds $n \times \text{MAD}$ simultaneously, where n acts as a scaling factor to determine the threshold for detection. This deviation must also persist for at least t minutes to ensure the anomaly is not caused by short-lived fluctuations. Based on heuristic analysis, t was set to 6 minutes in both our previous work [11] and this study.

It must be noted that the value of n plays a crucial role in determining the sensitivity of the anomaly detection system. In our previous work, n was fixed at 3, which ensured consistent detection by prioritizing anomalies with significant deviations, as these were assumed to have a greater impact on network functionality. Subtle anomalies, which in theory posed less risk to operations, were often not flagged. While effective for the case studies examined in this research, it may not hold true for all types of anomalies. Additionally, the fixed threshold lacked the flexibility needed to account for the variability inherent in real-world BGP traffic, leading to limitations in balancing the detection of meaningful anomalies and filtering out noise.

To address this, we introduced a dynamic threshold that retains the relevance-based logic of the fixed approach but adapts n to the data's variability. In highly variable conditions, n increases to prevent minor fluctuations from being misclassified, while in more stable conditions, n decreases to capture subtle anomalies. This adaptive capability ensures the system can consistently identify significant deviations while maintaining sensitivity to subtle patterns, making it more robust and effective in diverse network environments.

To determine the optimal n, we designed a combined performance score to evaluate its effectiveness. We empirically evaluated n values ranging from 1 to 5, testing 500 points within this range. For each value of n, we calculated the F1-score and precision, and these metrics were combined using the following formula:

$$\text{Combined Score} = 0.6 \times \text{F1-Score} + 0.4 \times \text{Precision} \qquad (2)$$

The optimization process was iterative. For each tested n, the combined score was compared to the previous best. When an improvement was observed, n was updated, and the corresponding F1-score and precision values were recorded. This dynamic adjustment enabled the system to automatically identify the threshold that provided the most effective trade-off between F1-score and precision.

The combined score employs a 3:2 weight ratio, specifically designed to enhance the system's performance by balancing accurate anomaly detection while minimizing false positives. The rationale behind this weighting is that the F1-score, calculated as the harmonic mean of precision and recall, offers a balanced measure of detection performance, treating precision and recall equally. In real-world BGP anomaly detection scenarios, this equal weighting may not sufficiently address the impact of false positives, which can lead to operational inefficiencies such as unnecessary rerouting or the misallocation of resources to investigate non-existent threats. By explicitly incorporating precision as a weighted component, the system adds targeted emphasis on reducing false positives, ensuring it remains effective in identifying anomalies without compromising practical applicability.

Heuristic validation confirmed the effectiveness of this weighted approach. Across multiple scenarios, including CodeRed, Slammer, and Nimda, the 3:2 ratio consistently achieved higher F1-score and accuracy compared to alternative configurations, outperforming the 2:3 ratio in both metrics and the 1:1 ratio in F1-score. While 1:1 slightly improved precision (by 0.01), the 3:2 ratio also showed a marginal improvement of 0.01 in F1-score. Given that F1-score provides a more balanced evaluation of precision and recall, we prioritized it as the primary metric for optimizing anomaly detection. This weighting provided the best trade-off between effective anomaly detection and minimizing false positives, ensuring high recall while maintaining detection reliability.

The dynamic threshold is designed to adapt automatically to variations in network traffic, ensuring that it remains effective across different conditions without requiring manual recalibration. Since the threshold is derived from MAD's statistical properties, it inherently adjusts as data distributions shift, mitigating the risk of threshold drift over time. This adaptability allows MAD to provide consistent anomaly detection across diverse network environments.

Figures 1 and 2 present the Slammer event extracted from the RRC04 collector, as a visual example to illustrate how this enhanced method can significantly improve detection

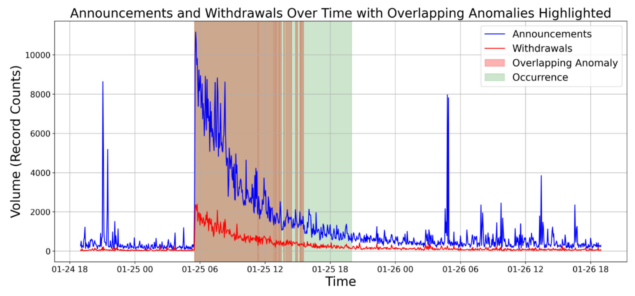


FIGURE 1. **MAD anomaly detection system with fixed threshold over Slammer Attack: visual representation of Announcements and Withdrawals volumes over time with anomalies marked in pink/brown and high-activity occurrence periods in green.**

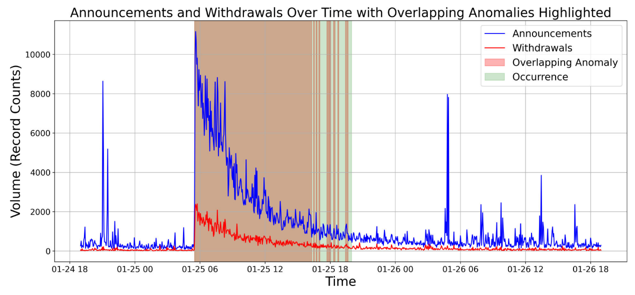


FIGURE 2. **MAD anomaly detection system with dynamic threshold over Slammer Attack: visual representation of Announcements and Withdrawals volumes over time with anomalies marked in pink/brown and high-activity occurrence periods in green.**

performance and increase accuracy. These visualizations provide clear evidence of the effectiveness of our dynamic threshold approach (see Fig. 2) compared to the previous fixed threshold method (see Fig. 1).

On the x-axis, we see the timestamp, and on the y-axis, we observe the volume of record counts. The blue line represents the number of announcements, while the red line indicates the number of withdrawals. The green highlighted box marks the actual occurrence period when these events were more prevalent, meaning that every instance within this timeframe is considered anomalous, whereas the white space is considered normal. Anomalies detected by our model, as previously explained, are highlighted in pink, which may appear brown when contrasted with the green box.

We can see the differences between the two figures. Figure 1) detects anomalies in the first three-quarters of the occurrence (green box), resulting in true positives but leaving a quarter as false negatives. In Fig. 2, the dynamic threshold adjusts to the data variability, allowing the statistical model to detect anomalies within the green box more effectively. This approach reduces false negatives without increasing false positives, demonstrating the enhanced performance and accuracy of the dynamic threshold method.

In Table 5, we present a comprehensive comparison of the MAD detection system's performance using a fixed threshold versus the dynamic threshold across all the anomalous events for RRC04. It is important to note that the same anomalous events were evaluated in both this research and our previous work. We evaluated the system's effectiveness by measuring F1-score, precision, and accuracy for each scenario. This analysis offers a clear understanding of the improvements achieved with the dynamic threshold approach, highlighting its enhanced capability to detect anomalies more accurately and reliably.

The MAD detection system with a fixed threshold achieved a precision of 0.95, accuracy of 0.91, and an F1-score of 0.69. The enhanced system, incorporating a dynamic threshold, improved performance with an accuracy of 0.93 and an F1-score of 0.77, while maintaining the same precision. These results demonstrate that the dynamic threshold effectively adapts to varying network conditions, making the enhanced MAD detection system more reliable for BGP anomaly detection.

D. DATA PROCESSING FOR ML IMPLEMENTATION

One of the main contributions of this study is evaluating how effective MAD is as a labeling method for machine learning models and neural networks. To achieve this, we compared MAD-labeled data with Occurrence-labeled data across different machine learning models. Occurrence-labeled data represents the time periods reported in the literature when anomalies were known to occur. While this approach provides a solid baseline, as it aligns with traditional practices in the field, it assumes that the entire duration of an event is anomalous, potentially including normal traffic within those intervals.

In contrast, MAD-labeled data is generated by the MAD detection system, treating all flagged instances as true positives. This approach focuses on detecting actual deviations in the data, offering a more dynamic and targeted labeling method. Our goal was to determine whether MAD's approach could provide more accurate and precise labels compared to occurrence-based methods. By comparing the two, we highlighted MAD's advantage in identifying true anomalies while excluding normal traffic during these periods, making it a potentially more effective labeling strategy.

To ensure the evaluation reflects real-world conditions, the creation of diverse datasets was essential. These datasets were designed to represent a wide range of anomalies, including CodeRed, Slammer, and Nimda worm attacks; the Telekom Malaysia incident; and the Moscow blackout. Each of these anomalies originates from different causes, such as malicious activity, misconfigurations, or link failures, impacting BGP traffic in distinct ways. This diversity ensures that the datasets comprehensively represent real-world challenges in anomaly detection, providing a robust foundation for assessing the effectiveness of the MAD labeling method.

To prepare the datasets for training, we removed the 'TIMESTAMP' column to focus on relevant features. Data was labeled as 1 for anomalous traffic (using MAD or occurrence-based methods) and 0 for normal traffic, resulting in three datasets: Occurrence Anomalous Dataset, MAD Anomalous Dataset and Normal Dataset.

TABLE 5. Comparison of MAD detection system performance: Fixed threshold vs. dynamic threshold.

Anomaly Event	Fixed Threshold			Dynamic Threshold		
	F1-score	Accuracy	Precision	F1-score	Accuracy	Precision
Code Red 1 v2	0.80	0.95	0.95	0.82	0.95	0.95
Slammer	0.76	0.89	1.00	0.90	0.95	1.00
Nimda	0.34	0.80	0.98	0.55	0.84	0.90
Moscow Blackout	0.72	0.96	0.88	0.70	0.96	0.94
TMnet	0.84	0.95	0.96	0.87	0.96	0.96

The anomalous datasets were then combined with the Normal Dataset to create two final datasets: Occurrence-based-labeling and MAD-based-labeling. This approach ensured consistency by including identical normal traffic in both datasets while maintaining their unique anomalous labels.

To address class imbalance, caused by the differing number of instances between the Occurrence-based-labeling and the MAD-based-labeling datasets we employed oversampling. We selected oversampling over undersampling based on a study by Mohammed et al. [20], which demonstrated that oversampling generally yields better performance across various machine learning classifiers. Therefore, we utilized the 'resample' function from Scikit-learn [21] to balance the datasets. This function operates by randomly sampling rows from the minority class with replacement, ensuring that the number of samples in each class matches the majority class. As a result, both datasets were expanded to a uniform size of 4,460 datapoints. This balancing process mitigates bias during training and testing by providing an equal representation of classes.

After labeling and balancing all the datasets, the data was split into training and testing sets with an 80−20 split, where 80% of the data was used for training and 20% for testing.

The flowchart in Fig. 3 provides a detailed step-by-step illustration of how the Median Absolute Deviation (MAD) detection system is integrated with machine learning models and the steps needed to do the comparative analysis. The steps within the red dashed line represent our main contribution, specifically the incorporation of MAD in ML models. Meanwhile, the steps outside the dashed line illustrate the comparative analysis. This analysis required an additional dataset to evaluate the performance between models trained with the MAD-based-labeling and the Occurrence-based-labeling datasets

The process begins with data collection, which sources information from public collectors or simple datasets. This is followed by data pre-processing, which includes feature extraction and dataset preparation. At this stage, we obtain two of our datasets: the Normal Dataset and the Occurrence Anomalous Dataset, as they do not require further processing. To obtain the MAD Anomalous Dataset, the next phase involves implementing the MAD detection system. This system identifies anomalies in the data, which are then labeled accordingly. Once all the datasets are created and labeled, we proceed to merge the datasets for the training and evaluation phase. In this step, we combine the Normal Dataset with both the MAD Anomalous Dataset and the Occurrence Anomalous Dataset to train separate machine learning models. The results from these training and evaluation step are then analyzed to determine the best performing model.

IV. MACHINE LEARNING MODELS

To evaluate the anomaly detection performance on both the Occurrence-based-labeling and MAD-based-labeling Dataset, we utilized an array of machine learning models known for their effectiveness on tabular data. Since we were working with tabular data and required quick decision-making capabilities, we focused on models that allow for fast inference. A key precedent for this decision comes from the work of Shwartz-Ziv and Armon [22], which demonstrated that when working with tabular data, classical methods like gradient boosting often perform as well as or even better than deep neural network approaches, while also offering faster training and inference times.

The chosen models consisted of Random Forest, ExtraTrees, XGBoost, LightGBM, and CatBoost. These models share a common foundation based on decision trees, with gradient boosting methods such as XGBoost which exhibit strong classification capabilities as documented in the work of Jayashinge et al. [23]. Additionally, we extended our analysis by including neural networks such as RNN, GRU, and LSTM to provide a point of comparison between traditional machine learning methods and deep learning approaches. This dual evaluation allowed us to explore the performance trade-offs and potential benefits of integrating MAD with different model types.

For this study, both the machine learning classifiers and neural network models were configured with specific hyperparameter settings to ensure optimal performance and replicability.

A. *HYPERPARAMETERS*

1) RANDOM FOREST

The algorithm's implementations in this work follows the heuristics employed in the Scikit-learn implementation [21]. This version uses the best split strategy. The entire training dataset was used to build each tree. The split criteria chosen for each tree was the Gini impurity, the trees were allowed to expand until all leaves were pure. The minimum number of samples required to split an internal node were set to 2. The

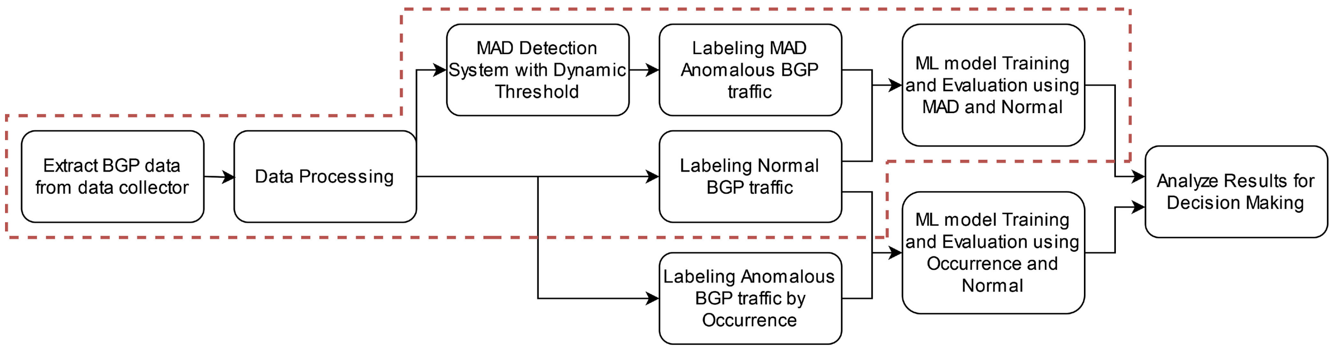

Extract BGP data from data collector
Data Processing
MAD Detection System with Dynamic Threshold
Labeling MAD Anomalous BGP traffic
Labeling Normal BGP traffic
Labeling Anomalous BGP traffic by Occurrence
ML model Training and Evaluation using MAD and Normal
ML model Training and Evaluation using Occurrence and Normal
Analyze Results for Decision Making

FIGURE 3. Flowchart illustrating the methodology for integrating the MAD detection system with machine learning models for BGP anomaly detection.

TABLE 6. Hyperparameters for random forest implementation.

Hyperparameter	Value/Description
Best Split Strategy	Best split strategy from sklearn
Training Dataset	Entire dataset used to build each tree
Split Criterion	Gini impurity
Tree Expansion	Trees expanded until all leaves were pure
Minimum Samples to Split Node	2
Minimum Samples at Leaf Node	1
Number of Estimators (Trees)	300

TABLE 7. Hyperparameters for extratrees classifier implementation.

Hyperparameter	Value/Description
Model Implementation	Scikit-learn ExtraTrees Classifier
Number of Estimators	100
Random State	42 (for experiment replication)
Other Parameters	Default settings from Scikit-learn

TABLE 8. Hyperparameters for XGBoost classifier implementation.

Hyperparameter	Value/Description
Model Implementation	XGBoost Python Library (XGBClassifier)
Label Encoder	Not used
Evaluation Metric	Log Loss (for binary classification)
Other Parameters	Default settings from XGBoost

minimum number of samples required to be at a leaf node was set to 1. Finally, 300 estimators were used to create the tree.

2) EXTRA TREES

We used the Scikit-learn implementation of the ExtraTrees classifier. The chosen hyper parameters consist of setting the number of estimators to 100 and setting the random state to 42, for reproducibility. The rest of the parameters were left in their default state in the Scikit-learn library.

3) XGBOOST

For our implementation, we used the XGBoost python library. We used the XGBClassifier model. The label encoder from XGBoost was not used and the chosen evaluation metric was log loss since the problem to address was a binary classification one. The rest of the parameters were left in their default state.

4) LIGHTGBM

We implemented LightGBM using its python based library. We set the model parameters to perform binary classification, setting up the boosting type to GBDT, using binary log

TABLE 9. Hyperparameters for LightGBM classifier implementation.

Hyperparameter	Value/Description
Model Implementation	LightGBM Python Library
Classification Type	Binary Classification
Boosting Type	Gradient Boosting Decision Tree (GBDT)
Evaluation Metric	Binary Log Loss
Learning Rate	0.05
Feature Fraction	0.9 (90% of training samples used per tree)
Number of Rounds	100 boosting rounds (trees)
Prediction Threshold	0.5 (above 0.5 indicates anomaly)

TABLE 10. Hyperparameters for CatBoost classifier implementation.

Hyperparameter	Value/Description
Model Implementation	CatBoost Python Library (Classification)
Number of Iterations	100 Gradient Boosting Iterations
Learning Rate	0.1
Tree Depth	6
Verbose Output	Information printed every 200 iterations
Data Handling	Training and testing datasets managed using Pools

loss as the evaluation metric, a learning rate of 0.05 and a 'feature fraction' of 0.9, indicating that we use 90% of the training samples randomly to train each tree. The model is trained for 100 rounds (boosting operations) for training the model. Each iteration adds one more tree to the ensemble. For performing predictions, the probabilities of belonging to each class are converted to binary labels using a threshold of 0.5, if a prediction is above 0.5 it is considered as an anomaly, if it is less, it is assumed to be representative of normal operation.

5) CATBOOST

We used the classification version of the model in Python. The model parameters were given the following values: it was set to 100 gradient boosting iterations, with a learning rate of 0.1, the tree was given a depth of 6, with information being printed every 200 iterations. The training and testing datasets were also handled using pools to put the data processing capabilities of the library to use.

6) RECURRENT NEURAL NETWORK (RNN)

We also used a RNN model implemented in PyTorch for binary classification. The model architecture consisted of a single RNN layer with a hidden size of 16 units, processing input sequences with one feature per time step. This architecture generated predictions for two classes. The

TABLE 11. Hyperparameters for RNN classifier implementation.

Hyperparameter	Value/Description
Model Implementation	PyTorch RNN Classifier
Classification Type	Binary Classification
RNN Type	Vanilla RNN
Number of Layers	1
Hidden Size	16
Input Size	1 (features per time step)
Output Size	2 (number of classes)
Learning Rate	0.01
Loss Function	CrossEntropyLoss
Optimizer	Adam
Batch Size	32
Number of Epochs	100
Learning Rate Scheduler	ReduceLROnPlateau

TABLE 12. Hyperparameters for LSTM classifier implementation.

Hyperparameter	Value/Description
Model Implementation	PyTorch LSTM Classifier
Classification Type	Binary Classification
RNN Type	Long Short-Term Memory (LSTM)
Number of Layers	1
Hidden Size	16
Input Size	1 (features per time step)
Output Size	2 (number of classes)
Learning Rate	0.01
Loss Function	CrossEntropyLoss
Optimizer	Adam
Batch Size	32
Number of Epochs	100
Learning Rate Scheduler	ReduceLROnPlateau (factor=0.1, patience=5)
Evaluation Metrics	Accuracy, Precision, Recall, F1 Score
Testing Batch Size	32
Learning Rate Tracking	Visualized per epoch

model was trained using the Adam optimizer with an initial learning rate of 0.01 and CrossEntropyLoss as the loss function. A ReduceLROnPlateau scheduler was employed to dynamically adjust the learning rate, reducing it by a factor of 0.1 after 5 epochs without improvement. Training was conducted over 100 epochs with a batch size of 32. Evaluation metrics included accuracy, precision, recall, and F1 score.The decision to use this configuration for the RNN was based on extensive experimentation, and is supported by previous research that reached similar conclusions [24].

7) LONG SHORT TERM MEMORY (LSTM)

We implemented an LSTM model using PyTorch. The model was designed for binary classification tasks, with a single LSTM layer and a hidden size of 16 units. The input consisted of sequences with one feature per time step, and the output layer provided predictions for two classes. We trained the model using the Adam optimizer with an initial learning rate of 0.01 and employed the CrossEntropyLoss as the loss function. A ReduceLROnPlateau scheduler was applied to adjust the learning rate dynamically, with a factor of 0.1 and a patience of 5 epochs. The model was trained for 100 epochs using a batch size of 32. Evaluation metrics included accuracy, precision, recall, and F1 score.

8) GATED RECURRENT UNIT (GRU)

We also implemented a GRU model in PyTorch for binary classification.The model architecture included a single GRU layer with a hidden size of 16 units, processing input sequences with one feature per time step. The output layer provided predictions for two classes. Training was performed

TABLE 13. Hyperparameters for GRU classifier implementation.

Hyperparameter	Value/Description
Model Implementation	PyTorch GRU Classifier
Classification Type	Binary Classification
RNN Type	Gated Recurrent Unit (GRU)
Number of Layers	1
Hidden Size	16
Input Size	1 (features per time step)
Output Size	2 (number of classes)
Learning Rate	0.01
Loss Function	CrossEntropyLoss
Optimizer	Adam
Batch Size	32
Number of Epochs	100
Learning Rate Scheduler	ReduceLROnPlateau (factor=0.1, patience=5)
Evaluation Metrics	Accuracy, Precision, Recall, F1 Score
Testing Batch Size	32
Learning Rate Tracking	Visualized per epoch

with the Adam optimizer, initialized at a learning rate of 0.01, and the loss function used was CrossEntropyLoss. To optimize the learning rate, we applied a ReduceLROnPlateau scheduler, which reduced the learning rate by a factor of 0.1 after 5 epochs of no improvement. The model was trained for 100 epochs using a batch size of 32. Evaluation metrics included accuracy, precision, recall, and F1 score.

V. RESULTS

In classification tasks, we used the following metrics. We looked at how the model classifies the data, comparing these classification results with the original labels—whether they are occurrence-based labels or MAD-based labels—provides us with four possible prediction categories:

- *True Positive (TP):* This category is assigned to a prediction when the model correctly predicts the positive class.
- *True Negative (TN):* This category corresponds to a model prediction that is correctly classified under the negative class.
- *False Positive (FP):* When a prediction is placed under this category, it means that the model erroneously classifies an instance of the negative class as positive.
- *False Negative (FN):* This category means that the prediction by the model incorrectly places a positive instance under the negative one.

Using these categories, it is possible to derive the required metric for evaluating the classification capabilities of each model:

- *Accuracy:* Accuracy corresponds to the ratio of correctly predicted instances with regard to the total instances present in the dataset. It is helpful as a general indicator of how often the model is correct. Since the dataset used for this task was balanced, accuracy is a reasonable metric as it is not affected by one class having more representation than another class.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

- *Precision:* Precision captures the ratio of correctly predicted positive instances to the total predicted positives.

It is useful as an indicator of false positive rate.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

- *Recall:* Recall represents the ratio of correctly predicted positive instances to all the instances in the actual positive class. A high recall indicates a low false negative rate.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5}$$

- *F1-score:* The F1-score is the harmonic mean of precision and recall. As a metric, it balances precision and recall, considering both false positives and false negatives. It is especially useful if the data is imbalanced.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

- *Area Under the Curve (AUC):* AUC measures the performance of the model by considering the area under the Receiver Operating Characteristic (ROC) curve. This curve plots the True Positive Rate (Recall) against the False Positive Rate (FPR). This metric provides a single scalar value representative of the likelihood of the model ranking a randomly chosen positive instance higher than a randomly chosen negative one.

In this work, we chose to report all these different metrics for each of the models used as means to demonstrate their classification capabilities.

A. GENERAL OBSERVATIONS

The performance of the models using both MAD-based-labeling and Occurrence-based-labeling datasets is summarized in Table 14. Across all machine learning classifiers—Random Forest, XGBoost, CatBoost, LightGBM, and ExtraTrees—the results showed significant improvements with MAD-labeled data. For the neural network-based classifiers, including the RNN, LSTM, and GRU models, the enhancements were less pronounced. While some neural network models demonstrated slight improvements, others maintained similar performance levels. On average, across the traditional machine learning models:

- *Precision* improved by 3.3%, reflecting a reduction in false positives.
- *Recall* improved by 3.7%, indicating better identification of true anomalies.
- *F1-score* increased by 3.2%, showing a balanced enhancement in both precision and recall.
- *Accuracy* improved by 3.6%, demonstrating overall better performance in anomaly classification.

Notably, 4 out of 5 MAD-labeled traditional classifiers achieved 1.00 in precision for the Normal class, as was the case for Random Forest, XGBoost, LightGBM, and ExtraTrees, indicating no false positives. This consistent enhancement across multiple models highlights the effectiveness of MAD labeling in improving classifier performance.

The neural network-based classifiers showed the following improvements:

- *Precision* decreased by 1%, meaning that the models increased the number of false positives in relation to the number of true positives.
- *Recall* improved by 3%, signaling a better identification of true anomalies.
- *F1-Score* saw overall no increments.
- *Accuracy* improved by 2%, which can be interpreted as an increase in correct predictions.

The neural network-based models experienced a small increase in false positives, as signaled by the decrease in precision. However, recall and accuracy also showed a slight increase. The results indicate that traditional machine learning-based methods saw a major benefit from the use of MAD-based labeling, while neural network-based methods became more sensitive at the cost of being less selective. Overall, traditional methods seem to be a better fit for anomaly detection, with their classification scores seeing general improvements after being trained and tested on MAD-labeled data.

B. TESTING MODEL PERFORMANCE ON NEW REGIONS AND UNSEEN ANOMALIES

To evaluate the generalization and robustness of the proposed model further, we conducted two additional experiments using data from previously unseen events.

The first experiment focused on detecting the AWS route leak of 2016 [25]. This evaluation tested the model's ability to recognize an anomaly it had never encountered before. Using data from RRC04, we ensured that the model identified the anomaly solely based on learned patterns without any prior knowledge of the event.

The second experiment assessed the model's ability to detect the TMnet anomaly in a different region. Initially, this anomaly was evaluated using data from European collectors (RRC00 and RRC04). To determine whether the model could adapt to different network environments, we tested its performance using data from RRC11 (New York, USA), another RIPE NCC collector, which operates in a geographically distinct region with different traffic characteristics.

Table 15 presents the performance metrics obtained for both experiments using models trained with MAD-labeled data.

For AWS Route Leak Detection, the Random Forest model trained with MAD labels achieved perfect detection, with 1.00 accuracy, precision, recall, and F1-score, demonstrating its ability to identify a novel anomaly.

For TMnet Detection in RRC11, the CatBoost model trained with MAD labels achieved the best performance, with an accuracy of 0.89 and an F1-score of 0.82.

In order to simulate a possible deployment scenario, where the model is continuously learning from novel data, in both experiments, the model was trained exclusively on normal data from the respective collectors before testing,

TABLE 14. Different classifiers' performance metrics (hybrid model).

ML Model	Class	Occurrence-based-labeling					MAD-based-labeling				
		Precision	Recall	F1-score	Acc	AUC	Precision	Recall	F1-score	Acc	AUC
Random Forest	0	0.96	0.96	0.96	0.96	0.99	1.00	0.98	0.99	0.99	1.00
	1	0.96	0.96	0.96			0.97	0.98	0.98		
XGBoost	0	0.98	0.95	0.96	0.96	0.99	1.00	0.99	0.99	0.99	1.00
	1	0.95	0.97	0.96			0.98	1.00	0.99		
CatBoost	0	0.94	0.91	0.93	0.93	0.97	0.99	0.98	0.98	0.98	1.00
	1	0.91	0.94	0.93			0.96	0.98	0.97		
LightGBM	0	0.96	0.94	0.95	0.95	0.99	1.00	0.98	0.99	0.99	1.00
	1	0.94	0.96	0.95			0.97	1.00	0.99		
ExtraTrees	0	0.97	0.95	0.97	0.96	1.00	1.00	0.99	0.99	0.99	1.00
	1	0.95	0.97	0.97			0.98	1.00	0.99		
RNN	0	0.86	0.89	0.88	0.88	0.93	0.99	0.86	0.92	0.90	0.97
	1	0.87	0.85	0.86			0.77	0.99	0.87		
LSTM	0	0.91	0.90	0.90	0.90	0.96	0.99	0.88	0.93	0.91	0.99
	1	0.89	0.91	0.90			0.80	0.98	0.88		
GRU	0	0.92	0.92	0.92	0.92	0.97	0.98	0.88	0.93	0.94	0.98
	1	0.91	0.92	0.92			0.80	0.96	0.87		

TABLE 15. Classifiers' performance metrics for evaluating model robustness on new regions and unseen anomalies.

ML Model	Class	AWS Route Leak					TMnet Anomaly				
		Precision	Recall	F1-score	Acc	AUC	Precision	Recall	F1-score	Acc	AUC
Random Forest	0	1.00	1.00	1.00	1.00	1.00	0.87	0.97	0.92	0.88	0.94
	1	1.00	1.00	1.00			0.92	0.71	0.80		
XGBoost	0	1.00	0.98	0.99	0.99	1.00	0.81	1.00	0.90	0.85	0.91
	1	0.96	1.00	0.98			1.00	0.54	0.70		
CatBoost	0	1.00	0.98	0.99	0.99	1.00	0.88	0.98	0.92	0.89	0.92
	1	0.96	1.00	0.98			0.94	0.73	0.82		
LightGBM	0	1.00	0.98	0.99	0.99	1.00	0.84	0.99	0.91	0.87	0.91
	1	0.96	1.00	0.98			0.97	0.62	0.76		
ExtraTrees	0	1.00	0.98	0.99	0.99	1.00	0.85	0.98	0.91	0.87	0.94
	1	0.96	1.00	0.98			0.93	0.67	0.78		
RNN	0	1.00	0.96	0.98	0.97	1.00	0.83	0.94	0.88	0.83	0.90
	1	0.93	1.00	0.96			0.83	0.62	0.71		
LSTM	0	1.00	0.98	0.99	0.99	1.00	0.85	0.94	0.89	0.85	0.90
	1	0.96	1.00	0.98			0.84	0.67	0.74		
GRU	0	1.00	0.94	0.97	0.96	1.00	0.86	0.96	0.91	0.87	0.93
	1	0.89	1.00	0.94			0.90	0.68	0.77		

meaning it had no prior exposure to any anomalies. This would evaluate the response of the model under real duress, and confirm if it could differentiate expected variations from anomalies based on each region's network conditions and time period. Two test datasets were created, one for each experiment, containing both normal and anomalous instances labeled according to their occurrence period.

VI. DISCUSSION

The Median Absolute Deviation (MAD) detection system effectively identifies variability-based BGP anomalies through its dynamic threshold adaptation, allowing it to detect sudden changes in routing behavior. However, because MAD relies primarily on announcements and withdrawals, its detection capacity is inherently limited. While these metrics are strongly correlated with BGP anomalies analyzed in this research, more complex anomalies, such as those involving intricate interactions between multiple features, evolving temporal patterns, or hidden correlations, may go undetected if they do not significantly affect these core indicators.

To overcome these limitations, the hybrid model integrates machine learning, enhancing the anomaly detection capability. By incorporating additional features such as AS-PATH lengths, duplicate withdrawals, and unique withdrawn prefixes, the model detects a broader range of anomalies that may not be evident through MAD alone. Machine learning excels at identifying intricate correlations and tracking feature evolution over time, providing a comprehensive framework for anomaly detection. This integration ensures adaptability while retaining efficiency, balancing the strengths of both approaches.

Our experimental results validate this hybrid framework. Traditional ML models trained on the MAD-labeled dataset achieved an overall accuracy of 0.99 and F1-score of 0.98, outperforming models trained on the occurrence-based dataset (accuracy and F1-score of 0.95). These findings underscore the value of MAD-labeled data in enhancing classification performance. While neural network-based models, including RNNs, LSTMs, and GRUs, were tested to provide a point of comparison, their performance improvements were less pronounced than those observed with traditional machine learning models. This highlights the practicality of using classical machine learning approaches for tabular data, especially in scenarios requiring fast inference and efficiency.

A comparison with existing hybrid techniques discussed in the Related Work section further demonstrates the advantages of our approach. While Peng et al.'s MAD-MulW model [9] achieved an average accuracy of 0.96, its reliance on sample expansion increased computational complexity. Similarly, Park et al. [10] achieved an F1-score of 0.99 and an AUC of 0.96 using an Autoencoder, but their method was limited to detecting a single type of anomaly, reducing its generalizability in real-world scenarios involving diverse anomalies. Moreover, many existing studies focus on individual anomaly types [6], [10] or evaluate models per event [2], [9], limiting their applicability in real-world contexts where a variety of anomalies coexist. These methods may excel at detecting specific events, such as worm attacks, but often fail to identify other anomaly types, like misconfigurations or link failures.

In contrast, our approach combines all analyzed anomalies into a single dataset, ensuring generalizability across diverse anomaly types. Although this is a challenging task–since different anomalies behave differently depending on their causes–they all significantly impact the network. To address this, the enhanced MAD system's dynamic threshold effectively adjusts to data variability, enabling the detection of various anomaly types, including indirect anomalies, link failures, and direct unintended anomalies. This capability enhances generalizability while maintaining high performance metrics and computational efficiency, making the approach practical for real-world applications.

To further validate the hybrid model's adaptability, we conducted additional experiments on an unseen anomaly and a new region, providing key insights into its effectiveness across different network environments. As previously discussed, MAD's reliance on announcements and withdrawals limits its ability to detect more complex anomalies that do not significantly impact these core indicators. Since the AWS Route Leak primarily involved a surge in announcement volume without corresponding withdrawals, MAD was unable to flag it as an anomaly. However, when using the hybrid approach, the Random Forest model trained with MAD labels successfully identified the event, achieving 1.00 accuracy, precision, recall, and F1-score. This confirms that the hybrid model does not merely replicate MAD's labeling but instead learns meaningful traffic patterns by analyzing a broader set of network attributes, including AS-PATH lengths, duplicate withdrawals, and unique withdrawn prefixes. This expanded feature analysis enables the model to capture intricate correlations and detect patterns that MAD alone may overlook, reinforcing its ability to identify complex anomalies while reducing the risk of confirmation bias.

In the TMnet anomaly experiment (RRC11, New York, USA), the hybrid model identified the anomaly with 0.89 accuracy, while MAD alone achieved 0.96 accuracy, demonstrating its adaptability to statistical deviations across regions. While MAD successfully detected this anomaly, its approach remains limited to variability-based behaviors, meaning it may overlook more complex anomalies. The hybrid model, despite being trained on normal traffic from RRC11, lacked exposure to anomalous traffic from that region, making detection more challenging.

While 0.89 represents a strong performance for a new region evaluation, it is important to discuss certain aspects of our evaluation methodology for these experiments. Our approach relies on occurrence times recorded in the literature, which may not always accurately reflect an anomaly's true impact across different regions. For instance, an anomaly reported as lasting two hours may have only affected a distant collector for 30 minutes due to propagation delays, differences in routing policies, or proactive mitigation efforts. Consequently, while the hybrid model may correctly detect the anomaly within its actual regional impact window, its measured performance may appear lower when evaluated over the full recorded duration. We highlight this limitation, as this evaluation method may not provide an equal advantage to all regions.

As shown, the hybrid model successfully detected an unseen anomaly, confirming its ability to generalize beyond known attack patterns. At the same time, MAD remains essential for zero-day attack detection, leveraging its adaptive statistical approach to identify emerging threats. Its ability to automatically label data without requiring manual intervention significantly improves efficiency and scalability. This feature ensures the system remains updated and effective in a constantly changing threat landscape.

These findings underscore the complementary strengths of MAD and ML models, demonstrating that both approaches are most effective when combined. While MAD alone provides a lightweight statistical foundation, ML models extends detection capabilities by learning complex traffic behaviors that MAD alone may not capture. By integrating MAD's statistical adaptability with the ML model's learning capacity, this hybrid framework offers a cost-effective, scalable solution for proactive BGP anomaly detection.

VII. CONCLUSION AND FUTURE WORK

This study demonstrates that integrating the Median Absolute Deviation (MAD) detection system with traditional machine learning (ML) models significantly enhances BGP anomaly detection, outperforming neural network-based models trained on the same data. Furthermore, implementing a dynamic threshold based on F1-score and precision values optimizes detection, improving the system's adaptability to varying network conditions. By leveraging MAD's dynamic threshold for labeling and expanding feature analysis through machine learning, the hybrid framework achieves both efficiency and high accuracy. Evaluating well-documented events such as CodeRed, Slammer, Nimda, the Moscow blackout, and the Telekom Malaysia misconfiguration yielded an overall accuracy of 0.99 and an F1-score of 0.98, surpassing existing methods in accuracy, generalizability, and computational efficiency.

Additional experiments further validated the hybrid model's adaptability, confirming its ability to detect an unseen anomaly (AWS Route Leak) and perform effectively in a new region (RRC11, New York, USA). These findings highlight the importance of testing models in diverse network environments to assess their robustness against previously unseen threats. However, the experiments also revealed challenges in regional adaptation, as traffic behaviors vary due to routing policies, infrastructure, and network topology. While the hybrid model performed well, these variations emphasize the need for further optimization to improve generalization across regions and ensure a broader applicability to less common anomalies.

Expanding validation across a broader range of events and datasets is essential for improving the model's generalization capabilities. Certain underrepresented regions and networks with fewer BGP peers limit the scope of anomaly detection. Future studies should incorporate additional routes and datasets from diverse geographic regions to ensure a more global perspective. Additionally, detecting less frequent anomalies remains a challenge due to uncertainties in occurrence times and the lack of precise start and end dates, complicating the creation of reliable labels for comparing occurrence-based and MAD-labeled datasets. Addressing these limitations could involve developing synthetic datasets to simulate zero-day anomalies and rare events, allowing the model to be tested in controlled environments for adaptability and robustness. Collaborating with organizations to access anonymized real-world data under strict privacy agreements would also help validate the model's effectiveness on unseen anomalies and underrepresented conditions.

Enabling the system for real-time anomaly detection remains a critical goal, as immediate identification and response to BGP anomalies are essential. However, implementing real-time solutions is challenging due to the restricted availability of live ISP data, often limited by confidentiality agreements and privacy concerns. Collaborating with network operators to access anonymized, real-time data or deploying the system in controlled environments could provide further validation of its practical applicability. Another potential development involves incorporating additional BGP message types, such as open and notification messages, into the detection framework. While announcements and withdrawals are strong indicators of variability-based anomalies, analyzing other message types may uncover subtler patterns or overlooked anomalies, further expanding the system's detection capabilities.

Finally, fostering open collaboration with the research community will be key to validating the system across evolving BGP threats. By continuously refining the integration between MAD and machine learning, future iterations of the hybrid framework can further enhance adaptability, ensuring effective and scalable BGP anomaly detection in dynamic network environments. These efforts will collectively advance the field of BGP anomaly detection, promoting proactive, efficient, and scalable solutions to protect critical Internet infrastructure.

ACKNOWLEDGMENT

Jose Antonio Cantoral-Ceballos would like to thank NVIDIA for GPU donation through the Academic Hardware Grant program.

DATA AVAILABILITY STATEMENT

The data presented in the study are openly available in the Routing Information Service (RIS) from the Réseaux IP Européens Network Coordination Centre (RIPE NCC) at https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/ (accessed on 25 June 2024).

REFERENCES

[1] B. Al-Musawi, "Detection of BGP anomalies using machine learning," Ph.D. dissertation, School Softw. Electr. Eng., Swinburne Univ. Technol., Melbourne, VIC, Australia, 2019. Accessed: Mar. 24, 2025. [Online]. Available: https://researchbank.swinburne.edu.au/file/627b88ea-e0d7-477a-9b64-6317fea582f7/1/bahaa_al_musawi_thesis.pdf

[2] Z. Li, A. L. G. Rios, and L. Trajković, "Detecting Internet worms, ransomware, and blackouts using recurrent neural networks," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, 2020, pp. 2165–2172.

[3] G. Huston, M. Rossi, and G. Armitage, "Securing BGP—A literature survey," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 2, pp. 199–222, 2nd Quart., 2011.

[4] *Rogers Outage Caused by Human Error, Outdated System, Probe Finds*, Can. Broadcast. Corp., Ottawa, ON, Canada, 2023, Accessed: Jan. 13, 2025.

[5] R. Copstein and N. Zincir-Heywood, "Temporal representations for detecting BGP blackjack attacks," in *Proc. 16th Int. Conf. Netw. Service Manag. (CNSM)*, 2020, pp. 1–7.

[6] O. R. Sanchez, S. Ferlin, C. Pelsser, and R. Bush, "Comparing machine learning algorithms for BGP anomaly detection using graph features," in *Proc. 3rd ACM CoEXT Workshop Big Data, Mach. Learn. Artif. Intell. Data Commun. Netw.*, 2019, pp. 35–41.

[7] H. Latif, J. Paillissé, J. Yang, A. Cabello, and P. Barlet-Ros, "Unveiling the potential of graph neural networks for BGP anomaly detection," in *Proc. 1st Int. Workshop Graph Neural Netw. (GNNet)*, 2022, pp. 7–12.

[8] K. McGlynn, H. B. Acharya, and M. Kwon, "Detecting BGP route anomalies with deep learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2019, pp. 1039–1040.

[9] S. Peng et al., "MAD-MulW: A multi-window anomaly detection framework for BGP security events," 2023, *arXiv:2312.11225*.

[10] H. Park, K. Kim, D. Shin, and D. Shin, “BGP dataset-based malicious user activity detection using machine learning,” *Information*, vol. 14, no. 9, p. 501, 2023.

[11] M. A. Romo-Chavero, J. A. Cantoral-Ceballos, J. A. Pérez-Díaz, and C. Martinez-Cagnazzo, “Median absolute deviation for BGP anomaly detection,” *Future Internet*, vol. 16, no. 5, p. 146, Apr. 2024.

[12] B. Al-Musawi, P. Branch, and G. Armitage, “BGP anomaly detection techniques: A survey,” *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 377–396, 1st Quart., 2017.

[13] G. Pang, C. Shen, L. Cao, and A. Van Den Hengel, “Deep learning for anomaly detection: A review,” *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–38, Mar. 2021.

[14] M. T. Almuqati, F. Sidi, S. N. M. Rum, M. Zolkepli, and I. Ishak, “Challenges in supervised and unsupervised learning: A comprehensive overview,” *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 14, pp. 1449–1455, Jul. 2024.

[15] (Reseaux IP Eur. Netw. Coord. Center, Amsterdam, The Netherlands). *Ripe Network Coordination Centre*, Jun. 2015, Accessed: Feb. 8, 2024. [Online]. Available: http://www.ripe.net/

[16] (Cent. Appl. Internet Data Anal., La Jolla, CA, USA). *Pybgpstream API Documentation*, Accessed: Apr. 10, 2023. [Online]. Available: https://bgpstream.caida.org/docs/api/pybgpstream/pybgpstream.html

[17] T. Arai, K. Nakano, and B. Chakraborty, “Selection of effective features for BGP anomaly detection,” in *Proc. IEEE 10th Int. Conf. Awareness Sci. Technol. (iCAST)*, 2019, pp. 1–6.

[18] A. Al-Bakaa and B. Al-Musawi, “A new intrusion detection system based on using non-linear statistical analysis and features selection techniques,” *Comput. Secur.*, vol. 122, Nov. 2022, Art. no. 102906.

[19] T. Pham-Gia and T. L. Hung, “The mean and median absolute deviations,” *Math. Comput. Model.*, vol. 34, nos. 7–8, pp. 921–936, 2001.

[20] R. Mohammed, J. Rawashdeh, and M. Abdullah, “Machine learning with oversampling and undersampling techniques: Overview study and experimental results,” in *Proc. 11th Int. Conf. Inf. Commun. Syst. (ICICS)*, 2020, pp. 243–248.

[21] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.

[22] R. Shwartz-Ziv and A. Armon, “Tabular data: Deep learning is not all you need,” *Inf. Fusion*, vol. 81, pp. 84–90, May 2022.

[23] M. Jayasinghe, S. Helmini, N. Jihan, K. Hewa, and S. Perera, “A machine learning based approach for predicting the performance of highly-concurrent server applications,” in *Proc. 18th Int. Conf. High Perform. Comput. Simul.*, 2021, pp. 1–8.

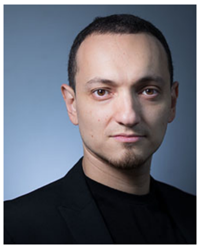
[24] J. G. Almaraz-Rivera, J. A. Perez-Diaz, and J. A. Cantoral-Ceballos, “Transport and application layer DDoS attacks detection to IoT devices by using machine learning and deep learning models,” *Sensors*, vol. 22, no. 9, p. 3367, 2022.

[25] Y. Xu. “NANOG 68: Decoding performance data from Internet outages.” Nanog, Nov. 2016, Accessed: Feb. 27, 2025. [Online]. Available: https://www.thousandeyes.com/blog/nanog-68-decoding-performance-data-internet-outages

MARIA ANDREA ROMO-CHAVERO received the B.S. degree in innovation and development engineering, with specializations in product development and advanced manufacturing and prototyping from the Instituto Tecnológico y de Estudios Superiores de Monterrey (ITESM), Santiago de Querétaro, Mexico, in 2022, and the M.S. degree in computer science from the Instituto Tecnológico y de Estudios Superiores de Monterrey, Monterrey, Mexico, in 2024.

In 2022, she conducted research to enhance the physical properties of aluminum 6061 surfaces by electrodeposition of multiwalled carbon nanotubes with ITESM. In 2021, she was awarded the Mitacs Fellowship for research on data analysis to develop a new bio-composite material capable of withstanding extreme weather conditions, in collaboration with the Composite Research Network and The University of British Columbia, Kelowna, Canada.

GUSTAVO DE LOS RÍOS ALATORRE received the B.S. degree in information technologies and the M.S. degree in computer science from the Instituto Tecnológico y de Estudios Superiores de Monterrey, Monterrey, Mexico, in 2022 and 2024, respectively, where he has been a Researcher with the Institute for the Future of Education since July 2024. His current research interests include robotics, machine learning, and natural language processing. He previously presented the paper “Understanding and Tracking of Deformable Objects in the Context of Humanoid Robotics: A Literature Review” at the 22nd Mexican International Conference on Artificial Intelligence in 2023. In 2023, he also participated in the Eighth International Congress on Information and Communication Technology with the article “Holistic Tweet-Based Sentiment Analysis on the China-Taiwan Conflict.”



JOSE ANTONIO CANTORAL-CEBALLOS (Member, IEEE) received the B.S. degree (Hons.) in electronic and communications engineering from the Tecnologico de Monterrey, Santiago de Querétaro, Mexico, in 2005, and the M.S. (Hons.) and Ph.D. degrees from the University of Manchester, Manchester, U.K., in 2009 and 2013, respectively, with a focus on algorithms for tomography imaging in embedded systems.

From 2013 to 2014, he was a Postdoctoral Research Associate at the University of Manchester, where he worked on the design and development of tomography data acquisition systems, and imaging algorithms from limited views. From 2015 to 2020, he was with the Advanced Technology Center (CIATEQ A.C.), Mexico, where his research efforts focused on the study of novel deep learning methods to solve digital signal processing and energy problems. He is currently a Research Professor with the Department of Computer Science, Tecnologico de Monterrey, where he continues his research on deep learning solutions to different problems, in particular to the study of neurological signals.

Dr. Cantoral-Ceballos was a recipient of the National Instruments Prize for his M.S. dissertation in 2009, and the Best Technical Presentation Award at the Seventh World Congress of Industrial Process Tomography, Krakow, in 2013.

JESÚS ARTURO PÉREZ-DÍAZ received the B.Sc. degree in computer science from the Autonomous University of Aguascalientes in 1995, and the Ph.D. degree in new advances in computer science systems from the Universidad de Oviedo in 2000, where he was a Full Associate Professor from 2000 to 2002. He is currently a Researcher and a Professor with the Tecnológico de Monterrey–Campus Guadalajara, Mexico, and a member of Mexican National Researchers Systems. He received a research stay with Louvain Le Nouveau University, Belgium. His research interests include cyber-security in SDN and multifactor authentication, where he has supervised several master’s and Ph.D. theses and published several articles in international journals. He was recognized by the COIMBRA Group as one of the Best Young Latin-American Researcher in 2006. He has been awarded by CIGRE and Intel for the development of innovative systems. He received the Best Student Award from the Autonomous University of Aguascalientes, for his B.Sc. degree.

CARLOS MARTINEZ-CAGNAZZO (Member, IEEE) received the B.Sc. degree in electrical engineering from the Universidad de la Republica, Montevideo, Uruguay, in 1998. He has worked in the internet industry in different roles, including large network design and network security. He is currently the Chief Technical Officer with Latin American Network Address Registry. His research interests include internet measurements, inter-domain routing, BGP security, IPv6, and DNS. He has taught courses on computer networking and cybersecurity, and partnered in different research projects trying to bridge the gap between academia and the industry.