A Dual-Model Anomaly Detection Algorithm for non-linear stream data in Smart City Environments

Anthony J. Bustamante*‡, Sarah Asad*, Daniela Nicklas†, Brent Lagesse*
*University of Washington, Bothell
Email: absuarez@uw.edu, sasad23@uw.edu, lagesse@uw.edu
†Otto-Friedrich-University Bamberg
Email: daniela.nicklsa@uni-bamberg.de
‡Corresponding author: absuarez@uw.edu

***Abstract*—In this paper, we introduce a novel approach for anomaly detection tailored to smart city infrastructures, utilizing a combination of regression algorithms. Our methodology employs two distinct regression models to generate future predictions from a given dataset. The primary model is crafted to yield high-fidelity predictions, while the secondary model is purposefully designed to introduce a degree of noise. Both models work together as a defense against DoS and Flooding attacks (or other attacks of the same nature) through the detection of abnormal levels of data inflow (detection of outliers). We calculate the alignment cost, or Euclidean distance, between the predictions from these two models, establishing a threshold against which real future traffic can be evaluated. The alignment cost or euclidean distance of the actual traffic is computed in relation to the high-quality predictions and then compared with the established threshold to pinpoint anomalies. Through experimentation with various regression algorithms, including linear regression, support vector regression, decision trees, etc., we identified an optimal combination for peak performance. Our evaluations, rooted in smart city datasets, emphasize the strength and consistency of this dual-model approach which outperformed traditional methods. Conclusively, the dual-model anomaly detection framework we propose stands out as an invaluable tool in defending smart city infrastructures from data irregularities and potential threats, highlighting the criticality of bespoke solutions in contemporary urban digital environments.**

***Index Terms*—Security, Flooding attacks, Denial of Service Attacks, Anomaly Detection, Pattern Recognition, Outlier Detection.**

I. Introduction

The emergence of smart cities—urban environments enriched with interconnected devices, sensors, and advanced information systems [1] — promises transformative changes in urban life and governance [2]. Such ecosystems, however, bring forth a plethora of data that necessitates rigorous monitoring to ensure data integrity and security [3].

Flooding attacks and anomalies in traffic in smart city environments exemplify critical challenges that can severely compromise the entire urban digital infrastructure [4]. Anomalies, if undetected, can lead to serious repercussions, affecting the decision-making processes, operational efficiency, and even public safety [5]. While a variety of techniques have been proposed for anomaly detection [6], smart city-specific datasets with their unique characteristics [7] and high data volume pose significant challenges to these traditional methods [8]. In addition, contingent on smart city infrastructure, tailored techniques are required for detecting anomalies [9].

To address this pressing matter, we propose a novel technique to detect outliers: a dual-model regression-based approach tailored specifically for flooding and DoS attacks in smart city data streams (however, the same proposed model could be used for the detection of outliers in some other fields with time series and non-linear nature) [10], [11].

Past approaches to anomaly detection, particularly for time-series data, have often been limited in their adaptability and sensitivity, especially when applied to intricate and voluminous smart city datasets [6], [12]. Our dual-model system, by contrast, has been designed with adaptability at its core, ensuring robust detection capabilities across diverse urban configurations [13].

Lastly, ensuring the security of data streams within smart cities is not merely a technical necessity but a critical factor in realizing the full potential of urban digital transformations [14]. Our proposed dual-model anomaly detection system stands at the forefront of efforts to build secure, reliable, and resilient smart city infrastructures [2], [15]. Such endeavors are paramount for harnessing the true potential of urban digitization, making cities safer, more efficient, and more responsive to their inhabitants' needs [16], [17].

II. Related Work

Anomaly detection in time series data and non-linear functions is a problem continuously studied with applications in various fields beyond Smart cities, including finance, manufacturing, healthcare, logistics, and industrial automation [18] [19]. Existing approaches in the literature can be broadly categorized into several methods [20], [21].

1) **Statistical Approaches** have been a foundation for anomaly detection in various domains [22].
2) **Machine Learning** Approaches have emerged as a powerful tool in understanding and identifying anomalies, especially in the domain of networking and system monitoring [23].
3) **Deep Learning Approaches** have seen a rise due to their capabilities in handling vast amounts of data and

the intricate patterns they can model [20]. For instance, GANs have been employed for anomaly detection in time series data [24], [25].

4) **Regression-based Approaches** have shown potential, especially when dealing with Wi-Fi signals and device detection challenges [12].
5) **Hybrid Approaches** combine the strengths of various algorithms to offer enhanced detection capabilities [21], [26].

Our proposed approach is a new methodology complementary to the existing methods that can be used to detect outliers by combining two regression algorithms with different characteristics to establish a robust threshold for anomaly detection.

III. Problem Statement and Solution

Detecting anomalies in time series data and non-linear functions, especially those commonly observed in smart cities, poses significant challenges. The dynamic nature of this data, intertwined with noise and prevalent non-linear patterns, exacerbates these challenges [26]. Traditional statistical and machine learning techniques for anomaly detection often struggle to adapt to evolving patterns, handle noisy data efficiently, or require complex feature engineering [8]. While regression-based methods have potential in predictive modeling for such scenarios, they are notoriously vulnerable to noise and outliers [12].

In our proposed solution, we advocate for a unique application of regression algorithms for anomaly detection in time series and non-linear functions. We generate two distinct predictions on future data behaviors using two separate regression algorithms:

- P_{best}: Derived from our top-tier regression algorithm, this prediction is optimized for accuracy.
- $P_{\text{reference}}$: Sourced from our secondary regression algorithm, this prediction intentionally produces slightly noisier results. If necessary, we can introduce additional noise to expand our anomaly detection threshold.

The central tenet of our approach is that the alignment cost or Euclidean distance (either through Dynamic Time Warping (DTW) or standard Euclidean measures) between the two predictions (P_{best} and $P_{\text{reference}}$) provides a spectrum within which we anticipate the future real traffic to align.

To evaluate the real traffic, denoted by T_{real}, we compute the alignment cost or Euclidean distance between T_{real} and P_{best}. The crux of our methodology is the assumption that this computed alignment cost or distance should fall within the threshold delineated by the two predictions. If the real traffic deviates beyond this boundary, it signifies an anomaly.

Our method offers the following advantages:

1) A merger of the predictive capabilities of regression models with the adaptability of a threshold-centric approach.
2) A significant reduction in the susceptibility to noise and outliers, a common issue with regression-based methods. This is achieved by incorporating a noise-tolerant reference prediction.
3) The inherent flexibility of the method makes it compatible with any regression algorithm(using batch learning or online learning), thus ensuring its applicability across diverse datasets and use cases.

IV. Methodology

A. Data Pre-processing

Time series and non-linear datasets inherently contain complexities that demand careful preprocessing. These temporal datasets present distinct temporal dependencies crucial for effective analysis. In our study, dates (time variable) serve as independent variables, with a counter value or count as the dependent variable (as shown in Table I). Proper preprocessing of these datasets is paramount, influencing the efficacy of regression models and the precision of anomaly detection.

The fundamental steps of our pre-processing method are: Date Parsing, Resampling, Handling Missing Values, Stationarity, and Feature Engineering.

Our approach bins the timestamps on data into 5-minute windows. Additionally, it is worth noting that we initially used a 5-minute window as it served as a foundational aspect of our project, primarily due to the need for consistency across various project components. However, this can be modified according to the user's needs. We also tested our approach with different window gaps, such as 1 minute and 2 minutes, and obtained the similar results.

B. Regression Algorithms

This study focuses on two regression algorithms that produced the best results for our use case: **Decision Tree Regressor (DTR)** and **Random Forest Regressor (RFR)** [27]. However, we tried multiple regression algorithms with different techniques, including batch learning and online learning. In our case, DTR and RFR were the models that performed the best (using batch learning). Our approach is not specifically limited to these regression algorithms and other regressors can be plugged in as needed for different applications or as new regressors are devised.

C. Threshold Calculation

1) Dynamic Time Warping (DTW): Dynamic Time Warping (DTW) is a technique used for measuring similarity between two temporal sequences that may vary in time, intensity or speed. It calculates an alignment cost that represents the best alignment between two sequences regardless of their non-linear variations in time. Given two sequences $A = a_1, a_2, \ldots, a_n$ and $B = b_1, b_2, \ldots, b_m$, DTW computes the optimal alignment cost between A and B by finding the path through the cost matrix C that minimizes the cumulative distance. The cost matrix C is defined as [28]:

$$C_{i,j} = d(a_i, b_j) + \min\{C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1}\} \quad (1)$$

where $d(a_i, b_j)$ is the distance between elements a_i and b_j and is usually calculated using the Euclidean distance.

2) Euclidean Distance: Euclidean distance is a metric used to measure the similarity between two vectors. Given two vectors $A = a_1, a_2, \ldots, a_n$ and $B = b_1, b_2, \ldots, b_m$, the Euclidean distance between A and B is defined as [29]:

$$d_{euc}(A, B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \tag{2}$$

3) Threshold Calculation: In this study, we calculate thresholds using both the alignment cost (DTW) and the Euclidean distance. The goal is to establish a threshold that represents the expected distance between two future predictions: one that is the best possible prediction (Future Prediction 1) and another that serves as a reference point (Future Prediction 2). The reference prediction is preferably a bit noisy, allowing for more flexibility in the threshold.

Given two future predictions A and B, the alignment cost threshold is calculated as:

$$T_{ac} = \text{DTW}(A, B) \tag{3}$$

and the Euclidean distance threshold is calculated as:

$$T_{ed} = d_{euc}(A, B) \tag{4}$$

where $\text{DTW}(A, B)$ is the alignment cost between A and B, and $d_{euc}(A, B)$ is the Euclidean distance between A and B.

We also tested other algorithms to define a threshold; however, we did not achieve the results we were aiming for. Some of the algorithms we tested included: Cosine Similarity, Cross-Correlation, and Pearson Correlation Coefficient.

D. Anomaly Detection

Taking into account the equations in 3 and 4, we then analyze the real traffic by calculating the alignment cost or Euclidean distance between the real traffic and the best prediction (P_{best}). If the calculated distance is within the previously computed threshold, the real traffic is considered normal; otherwise, it is considered anomalous. Specifically, the real traffic C is considered normal if:

$$\text{DTW}(A, C) \leq T_{ac} \quad \text{or} \quad d_{euc}(A, C) \leq T_{ed} \tag{5}$$

where T_{ac} is the alignment cost threshold, and T_{ed} is the Euclidean distance threshold.

The threshold calculation approach was implemented as part of our experiment. Data can be sourced from various formats or directly collected in real-time. In our experiment, we analyze data both in real-time (running simulation by replaying real-world data) and from CSV files. The procedure involves merging future predictions with actual traffic data, addressing any data gaps, and aggregating the information into 5-minute intervals. Following this, the code computes the alignment cost or Euclidean distance between these sequences, establishing a threshold representing the expected distance between the predictions and real data.

V. Experiment Setup

In this section, we outline the experimental configuration employed to assess the efficiency of our proposed approach in identifying anomalies within the data stream from the smart city. (Figure 1). In the next sub-sections, we illustrate the deployed topology, data collection process, and the specifics of our anomaly detection implementation.

A. Deployment Topology

Our experimental setup, we were provided access to, a smart city environment, which currently consists of six Wi-Fi probing sensors that are deployed across the busiest areas of Bamberg, Germany (see Figure 1) [12]. The purpose of these sensors is to capture anonymized probe requests from Wi-Fi-enabled devices, generating real-time data that represents the overall network traffic in the city.

As shown in Figure 1, the Wi-Fi sensors communicate with a processing backend server using TCP. The backend server receives JSON-formatted requests from the sensors and implements our proposed anomaly detection algorithm. The processed data, either tagged as normal or anomalous, is then stored in a database. APIs can subsequently consume the stored data for additional processing.

B. Data Collection and Pre-processing

The Wi-Fi sensors collect probe requests, which are small data packets sent by Wi-Fi-enabled devices to discover nearby Wi-Fi networks as seen in Figure 1. These probe requests contain information such as the device's MAC address, timestamps, and additional details about the street and city, among other things. In our experimental setup, the sensors capture these probe requests and anonymize the data to protect individuals' privacy. The anonymized data is then encapsulated in JSON format before being sent to the backend server via TCP/HTTPS. The data included the following fields:

- **eventype**: Event defined in our sensors based on their position and status.
- **epocutc**: Timestamp about when the event took place.
- **zone**: Sensor's location in the city.
- **mac_address**: Hash of the MAC addresses of individuals, hashed using SHA-224.
- **RSSI**: RSSI value of each device in the vicinity.
- **techtype**: Technology type used for data collection. For this study, only WiFi receptors were employed.

At the backend server, the incoming JSON requests are parsed, and the relevant information is extracted for further processing. The data is then pre-processed, following the steps outlined in Section IV-A to prepare it for anomaly detection.

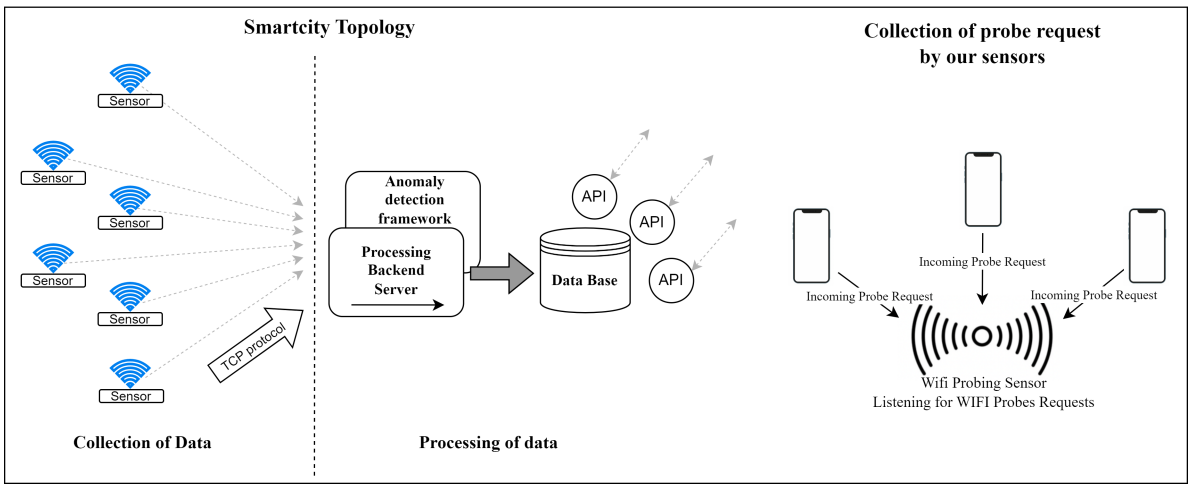

Fig. 1. Smart City Architecture Used For Our Experiments and Demonstration of WiFi Probe Reception by the Sensor

C. Anomaly Detection and Defense Mechanisms

Our anomaly detection algorithm, as presented in previous sections, is implemented on the backend server. It analyzes the incoming traffic, represented by probe request data from the Wi-Fi sensors, to identify any deviations from expected patterns. The primary focus of our experiment is to detect anomalies/outliers indicative of DoS, DDoS, flooding attacks, or other malicious activities of the same nature that could disrupt the city's network infrastructure.

If the algorithm detects an anomaly, the server can trigger predefined protection mechanisms, such as, another defense mechanism, alerting network administrators, blocking suspicious traffic, or adjusting network parameters to mitigate the impact of the attack. Conversely, if no anomalies are detected, the traffic is considered legitimate and allowed to proceed to the database, where APIs can access and consume the data for various applications. Note that since our algorithm works on both streaming data and datasets, the algorithm can be re-run on existing datasets if better regressors for a particular attack are developed.

This experimental setup is designed to validate the effectiveness of our proposed anomaly detection approach in a real-world smart city environment, emphasizing its potential to protect against malicious network activities.

VI. Evaluation and Results

A. Preprocessing and Feature Engineering

The dataset used in the study contained the fields: year, month, day, hour, minute, second, and counter. Interaction terms were introduced in the data to capture inter-feature relationships and improve model performance (Feature engineering part indicated in section IV-A). These interactions were obtained by multiplying two features together to create a new feature. A sample of the data after feature engineering is shown in Table II.

TABLE I
Sample data from the datasets

Year	Month	Day	Hour	Minute	Second	Counter
2023	1	30	12	33	2	2
2023	1	30	12	33	4	3
...	...	...	...	...	...	...
2023	7	10	0	0	8	1
2023	7	10	0	0	18	1

B. Model Training

Our model was trained on a 3 million entry dataset. We used an 80/20% split of training to testing data and selected the best performing models. For our research, we utilized the `scikit-learn` library [30], but also tried online-learning libraries like River [31]. Furthermore, the hyperparameters used for training our models were set to their default values in `scikit-learn` (except `number of estimators = 1` and `random_state = 0` for RFR). We experimented with various configurations, but in our specific scenario, they did not yield a substantial impact.

C. Threshold Generation

Algorithm 1 describes the threshold generation process.

The thresholds generated from the predictive models were used to evaluate real traffic data. They provided an effective means of identifying abnormal traffic patterns and alerting for potential flooding or DoS attacks. To calculate the euclidean distance or alignment cost for the real traffic we compared the real traffic data with the algorithm with best performance(P_{best}).

D. Problem to solve

Figure 2 illustrates the category of Non-linear Time Series challenges relevant to Smart Cities. The dataset spans 7 months, from January 2023 to July 2023. However, for the graph presented in this paper (Figure 2), we have chosen to depict data from just a single week.

TABLE II
SAMPLE DATA AFTER FEATURE ENGINEERING

Table I values(except counter)	Interactions						Counter
	month_day	day_hour	month_hour	hour_min	hour_sec	min_sec	
For our final dataset	month*day	day*hour	month*hour	hour*min	hour*sec	min*sec	2
we copied the same values from	1	84	12	550	120	1	3
Table I, except	...	...	...	...	...	...	...
the values in counter and year	1	30	72	451	1288	100	5

Algorithm 1 Threshold Generation

Require: reference data: ref_data, prediction data: $pred\_data$ or $real\_data$

Merge ref_data and $pred\_data$ or $real\_data$ on time columns
Fill NaN values in counts
Create a 'datetime' column from separate date and time columns
Define the frequency for the grouping: $freq \leftarrow' 5T'$
Initialize empty list: $results \leftarrow []$
for each group g in merged_data.groupby(pd.Grouper(freq=freq)) **do**
    Calculate ed or ac between real and predicted values in g
    Extract date-time details dt
    Append $(dt, ed$ or $ac)$ to $results$
end for
Save $results$ to a file, Memory, or DB

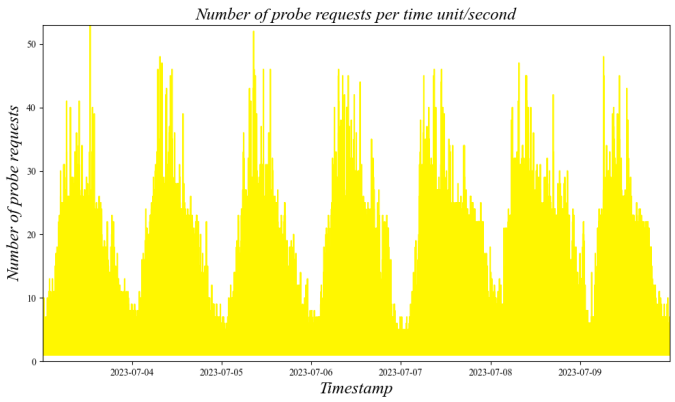

Fig. 2. One week example of the dataset.

E. Predictions vs real traffic

In Figure 3 we can see the plots for the real traffic that we are evaluating with our algorithm, the predictions with our best model (in red), which is almost identical to the real traffic (in blue), and lastly the noisy prediction with our second model that we used as a reference point to create our threshold(in purple).

In our dual-model approach, the secondary prediction was derived from what we discerned as the second-best regression model. To create a meaningful differential for establishing a threshold, we deliberately introduced a controlled amount of random noise. It is essential to note that during our experimental phase, we evaluated both the Euclidean distance and the alignment cost across a myriad of models, ultimately selecting those that demonstrated superior performance in the smart-city domain.

Given:

- P_1 as the prediction of the primary (best) model.
- P_2 as the prediction of the secondary (second-best) model before noise introduction.
- N as the random noise vector.
- ED as the Euclidean distance.
- AC as the alignment cost.

Note: The noise `N` introduced in this experiment was generated by multiplying the results of our second-best model with a random number ranging from 1 to 3.5, however, people can use any other type of technique to introduce noise with the objective of expanding the threshold that we want to produce for the detection of outliers.

The prediction with noise for the secondary model can be represented as:

$$P_2' = P_2 + N \tag{6}$$

The Euclidean distance and alignment cost can then be computed between P_1 and P_2' to determine anomalies:

$$ED(P_1, P_2') \tag{7}$$

$$AC(P_1, P_2') \tag{8}$$

Moreover, we also appraised the performance of our selected models using multiple evaluation metrics including the R^2 score [32], Mean Absolute Error (MAE) [33], Mean Squared Error (MSE) [34], and Root Mean Squared Error (RMSE) [35]. These metrics offer insights into the accuracy and reliability of our regression models.

The results from our evaluation are summarized in the table III:

TABLE III
SUMMARY OF REGRESSOR ERROR RESULTS

Evaluation methods	DTR	RFR (without noise)
R2	-0.809517785	-0.816383466
MAE	3.149929931	3.190064165
MSE	34.34288167	34.4731856
RMSE	5.860279999	5.871387025

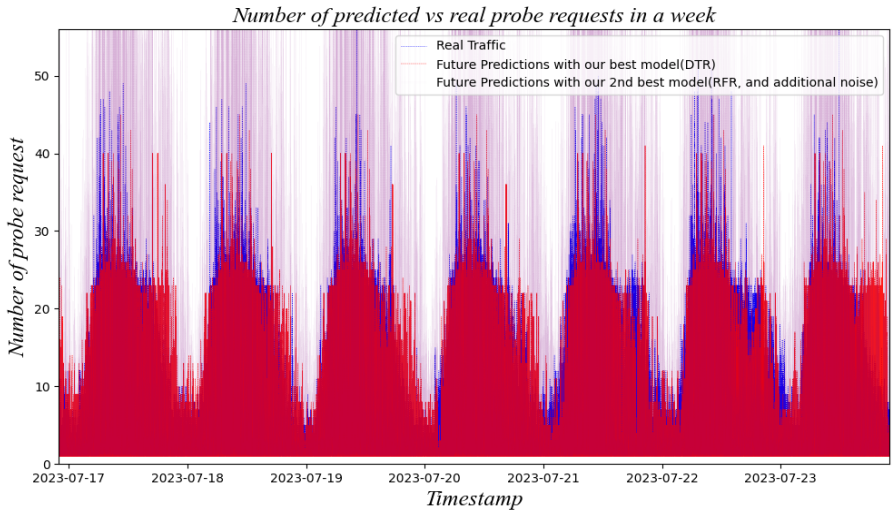

Fig. 3. Comparison for our best model vs reference point model vs real traffic

Upon examining the values presented in the table, we observe that the R^2 values for DTR marginally surpass those for RFR, and similar trends appear across MAE, MSE, and RMSE. The data stream emanating from smart city infrastructures is characteristically inundated with noise and exhibits non-linearity. Thus, achieving perfect evaluation scores can be elusive, especially in complex scenarios.

Furthermore, it's crucial to comprehend that these evaluation metrics, while offering insights into the model's performance, should be considered in conjunction with the specific nature of the data and the application's context. This observation resonates with findings in the literature, where the intricacies of dealing with noisy and non-linear data in networking-monitoring systems lead to less-than-perfect R^2 values [23] Our research emphasizes the significance of adaptive modeling in smart city environments, focusing not merely on achieving high scores but on building models that adeptly discern anomalies amidst the vastness of urban digital data streams.

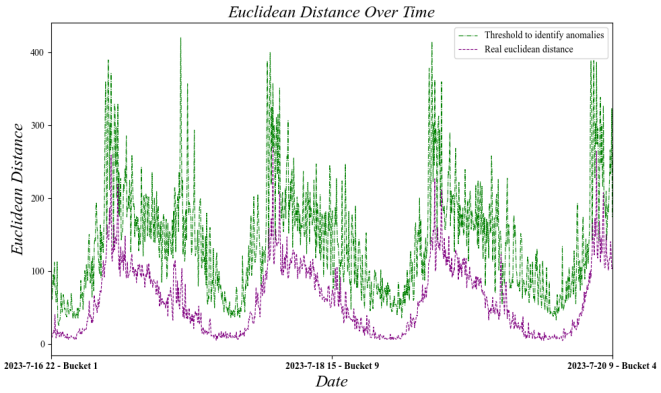

Fig. 4. Comparison of Established Euclidean Distance Threshold(green) to Actual Traffic Euclidean Distance(purple).

F. Analysis of Euclidean Distance Outcomes

Figure 4 delineates the derived threshold juxtaposed against the genuine Euclidean distance associated with real-world traffic. This congruence substantiates that our algorithmic approach is robust and aptly suitable for handling challenges inherent to time series data as well as nonlinear complexities (Characteristics of smart city traffic).

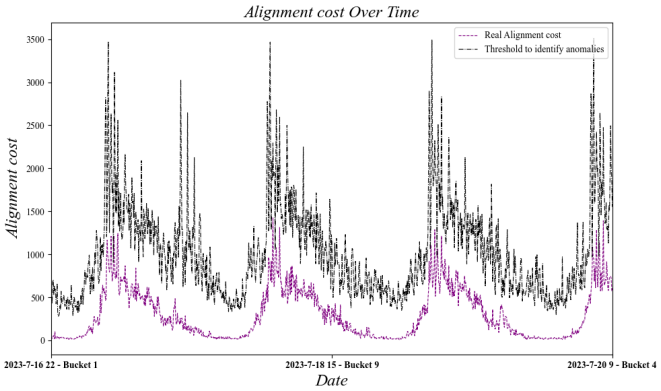

Fig. 5. Contrast between Projected Alignment Cost Threshold(black) and Actual Traffic Alignment Cost(purple)

G. Evaluation of Alignment Cost Metrics

Figure 5 juxtaposes the calculated alignment cost threshold with the actual alignment cost observed in real-world traffic. This visual representation further highlights the effectiveness of our proposed algorithmic approaches, showcasing their proficiency in handling time series challenges and nonlinear dynamics.

H. Benchmarking Proposed Methodology Against Statistical Techniques

Figure 6 displays the performance of various thresholds in relation to the true alignment cost. The distinct curves represent:

- The real alignment cost (dashed purple curve).
- The anomaly detection threshold derived from our proposed technique (solid black curve).
- An alternative threshold determined using the real statistics from the preceding day (brown dash-dot curve).
- A threshold established by averaging statistical values from the past week (magenta dotted curve).

Our evaluation also extended to setting thresholds based on statistical analysis. Specifically, we assessed the alignment cost (and, although not illustrated here, the Euclidean distance) between our premier model and the statistical outcomes from the day prior, as well as the cumulative averages from the last seven days. As Figure 6 elucidates, these statistically derived thresholds did not resonate optimally with the real-world alignment costs. This stark contrast accentuates the superior efficacy of our proposed approach in accurately mirroring the genuine alignment costs.

$$\tau_{proposed} = f_{regression}(data) \tag{9}$$

$$\tau_{day} = f_{statistical}(data_{day-1}) \tag{10}$$

$$\tau_{week} = \frac{1}{7}\sum_{i=1}^{7} f_{statistical}(data_{day-i}) \tag{11}$$

Where:

- $\tau_{proposed}$ represents the threshold from our proposed methodology.
- τ_{day} denotes the threshold derived from the prior day's statistics.
- τ_{week} symbolizes the threshold computed by averaging the last week's statistical data.

As we can see from Figure 6, our proposed method is more suitable than the other approaches, which is another way to demonstrate the value of our proposal.

I. Performance Analysis under Flooding Attack

Figure 7 showcases the system's capability to detect anomalies even under subtle adversarial conditions, such as a simulated flooding attack (this encompasses DoS, replay attacks, and DDoS attacks as well for our smart city environment). The alignment cost resulting from the fabricated "flooding attack data" significantly exceeds the threshold defined by our methodology. This contrast reaffirms the sensitivity and robustness of our anomaly detection mechanism. In algorithm 2 we show the attack model we used for our testing.

To simulate the attack, fake data was generated by dispatching random JSON queries at intervals ranging between 40 to 200 within a 5-minute span. This volume is markedly higher than the typical peak observed in legitimate traffic (approximately 50 queries). However, it is worth noting that in real-world attack scenarios, the volume of such queries could escalate into thousands. This underlines the superior sensitivity of our model: if it can accurately identify anomalies with such low-level surges, detecting larger-scale attacks becomes exponentially more straightforward.

Algorithm 2 Attacker Model and Flooding Attack Simulation

Context: The primary threat for our Smart city infrastructure is flooding attacks aiming to overwhelm sensors or backends by inundating the system with an excessive number of probe requests, often accompanied by forged MAC addresses.
Attack Techniques:
1. MAC Address Spoofing:
Craft random MAC addresses to impersonate numerous devices and camouflage malicious traffic amidst legitimate requests.
Require: Number of MACs to generate: N
for $i = 1$ to N **do**
    $MAC\_addr \leftarrow$ generateRandomMAC()
end for
2. Probe Request Flood:
Deploy algorithms from malicious devices (e.g., compromised laptops) to incessantly send probe requests, exhausting WiFi sensors' resources.
sendProbeRequest(MAC_addr)
3. Volume Amplification:
Require: Duration: D
while D not elapsed **do**
    $MAC\_addr\_list \leftarrow$ generateMultipleMACs()
    sendVolumeFlood(MAC_addr_list)
end while
4. Replay Attacks:
Eavesdrop and capture legitimate packets, then resend them to cause confusion and potential malfunctions.
if Replay attack enabled **then**
    $captured\_packets \leftarrow$ eavesdrop()
    resend($captured\_packets$)
end if

Mathematically, the surge in traffic can be represented as:

$$\Delta Q_{attack} = Q_{attack} - Q_{normal} \tag{12}$$

Where:

- Q_{attack} denotes the number of queries during the flooding attack.
- Q_{normal} signifies the standard query count in genuine traffic.

Given our observed values, we have: $\Delta Q_{attack} = 150$.

The modest value of ΔQ_{attack} and our model's ability to spot it elucidates its refined sensitivity. With the ever-evolving threat landscape, having an anomaly detection system with such sensitivity becomes paramount for maintaining robust security postures.

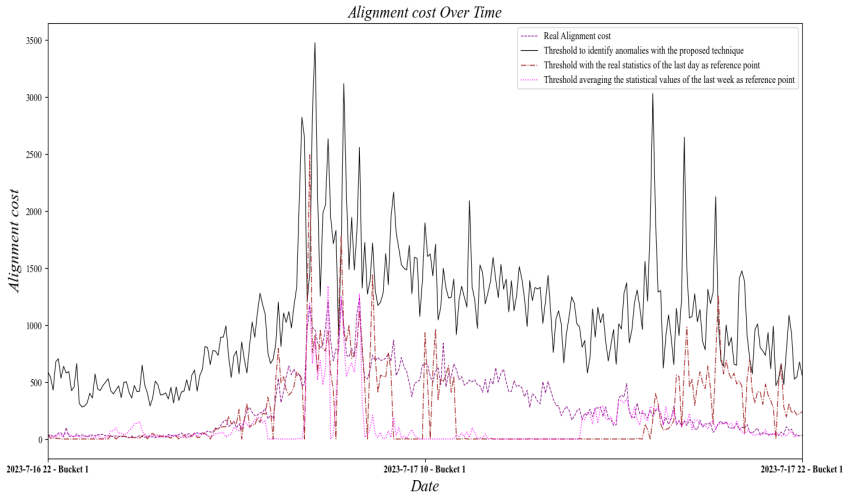

Fig. 6. Comparative Analysis: Proposed Model, Reference Point Model, and Real Traffic Alignment Cost

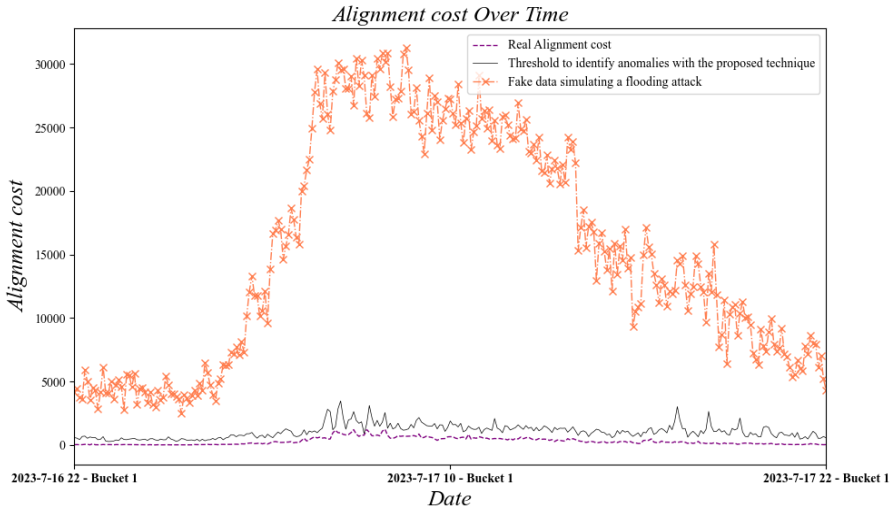

Fig. 7. Anomaly Detection Performance during a Simulated Flooding Attack

VII. Anomaly Detection Using the Anomaly Indicator

An essential aspect of anomaly detection, especially in complex environments like smart cities, is to have a quantifiable measure that can effectively differentiate between typical and anomalous data. To this end, we introduce an Anomaly Indicator (AI) given by the formula:

$$\text{AI} = \frac{\text{Threshold value}}{\text{Real value}} \tag{13}$$

The AI serves as a decisive metric in our analysis. When AI is greater than or equal to 1, it suggests that the observed traffic conforms to the expected patterns and does not exhibit any anomalies. Conversely, if AI is less than 1, it indicates the presence of anomalous behavior in the traffic data.

This straightforward yet effective metric provides a clear boundary that distinguishes between benign and malign data. By using AI, we can swiftly ascertain the nature of the traffic without delving into the intricacies of underlying patterns or dependencies.

From Figure 8, the calculated Anomaly Indicator (AI) values further emphasize the accuracy of our model in differentiating between benign and malign data. Specifically, when using the formula 13, the benign data (depicted in blue) consistently results in AI values closer to or greater than 1, indicating the absence of anomalies. In contrast, the malign data (depicted in coral) produces AI values substantially below 1, signifying the presence of anomalies. This clear distinction not only highlights the efficacy of our anomaly detection mechanism but also underscores its vital role in safeguarding smart city infrastructures. The model's consistent performance

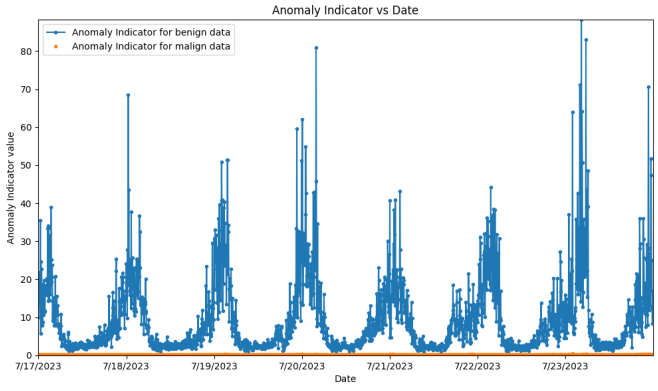

Fig. 8. Anomaly Indicator vs Date for Benign and Malign Data

across different data points further solidifies the robustness of our proposed approach.

Furthermore, during our experiments, we subjected our defense mechanism to a rigorous testing environment (simulation of our real environment). Over a span of 168 hours(one week), we dispatched approximately 80,640 probe requests to evaluate the system's robustness (We chose to use our euclidean distance results). The results, as Table IV shows, were commendable:

TABLE IV
PERFORMANCE METRICS OF THE DEFENSE MECHANISM

Metric	Score
Accuracy	100%
Precision	100%
Recall	100%
F1 Score	100%
True Positive Rate (TP)	100%
False Positive Rate (FP)	0%

While we are optimistic about the reproducibility and consistency of these results in real-world scenarios, we acknowledge the importance of comprehensive testing. Although the current outcomes are promising, we believe in leaving no stone unturned. As a part of our ongoing research, we are in the process of evaluating our defense mechanism against an even broader array of attack scenarios in our real infrastructure.

VIII. Discussion

Smart Cities' monitoring applications produce copious amounts of dynamic data. These data streams are often characterized by time series and exhibit non-linear patterns, making anomaly detection a significant challenge. The intricacies of urban data, such as fluctuations, noise, evolving non-linearities, and the sheer complexity, can render many conventional methods inadequate in a smart city setting. The approach presented in this paper provides a specialized solution for these challenges, capitalizing on the predictive capabilities of regression models coupled with the flexibility of a threshold-based method.

A prime advantage of our method is its adaptability. It presents a consistent framework suitable for navigating the diverse datasets typical of smart cities, irrespective of the regression algorithm employed. Moreover, our approach is designed to factor in noise when making predictions with the secondary model. This design inherently offers resilience against outliers, thus mitigating a prevalent shortcoming of conventional regression techniques.

For practical deployment in real-world scenarios, it's imperative to consider online or pseudo-online learning mechanisms that continually update the model. While online learning updates the model as new data arrives, our current study utilizes batch learning. This means our model undergoes retraining with each new data batch from the Wifi sensors. Future work will address these aspects, striving for a more seamless and real-time model update mechanism.

IX. Conclusion

In the intricate and dynamic data landscape of smart cities, our results underscore the dual-model regression's effectiveness in identifying anomalies. Compared to traditional benchmarks like prior day metrics or weekly averages, our method aligns better with actual urban traffic patterns, showcasing the strength and resilience of our approach for smart city data.

In this paper, we've adopted a standardized methodology to reproduce our results, using the best-performing model to compare with actual traffic and employing the second-best model as a reference. However, it's important to understand that the choice of the reference algorithm does not need to be strictly the second-best model. As long as the chosen reference model can produce an AC or EC that yields a useful threshold for outliers, it's a viable option. Similarly, while we used the top-performing model for our comparison, one could also consider other models that give accurate predictions, and better performance e.g. In essence, we've charted a structured path for achieving our results, but there's room for variation in the approach without compromising the objectives.

Lastly, this research has been oriented towards smart cities, its foundations could very likely be beneficial in a wider context in time series or non-linear dynamics. Thus, while the immediate focus has been on urban digital ecosystems, the potential universality of the approach beckons further exploration. Future endeavors could delve deeper into refining this methodology, gauging its performance across diverse datasets, and extending its horizons beyond smart cities to address broader challenges in time series and non-linear domains.

References

[1] M. Batty, K. W. Axhausen, F. Giannotti, A. Pozdnoukhov, A. Bazzani, M. Wachowicz, G. Ouzounis, and J. Portugali, "Smart cities of the future," *The European Physical Journal Special Topics*, vol. 214, no. 1, pp. 481–518, 2012.

[2] H. Chourabi, T. Nam, S. Walker, J. R. Gil-Garcia, S. Mellouli, K. Nahon, T. A. Pardo, and H. J. Scholl, "Understanding smart cities: An integrative framework," in *2012 45th Hawaii International Conference on System Sciences*. IEEE, 2012, pp. 2289–2297.

[3] R. Roman, J. Zhou, and J. Lopez, "Securing the internet of things," *Computer*, vol. 44, no. 9, pp. 51–58, 2011.

[4] Y. Zhang, N. Meratnia, and P. J. Havinga, "Flooding attack detection in smart cities," in *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*. IEEE, 2014, pp. 440–445.

[5] M. S. Hossain and G. Muhammad, "Anomaly detection in smart city using deep and machine learning models," *IEEE Access*, vol. 7, pp. 38 583–38 593, 2019.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[7] M. Batty, "Big data, smart cities and city planning," *Dialogues in human geography*, vol. 3, no. 3, pp. 274–279, 2013.

[8] F. Stoffel, F. Fischer, and D. A. Keim, "Finding anomalies in time-series using visual correlation for interactive root cause analysis," in *Proceedings of the Tenth Workshop on Visualization for Cyber Security*, ser. VizSec '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 65–72. [Online]. Available: https://doi.org/10.1145/2517957.2517966

[9] S. Gupta and S. C. Panda, "A survey of data-centric challenges in smart city communication," in *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*. IEEE, 2016, pp. 376–381.

[10] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[11] X. Zhang, C. Furtlehner, and M. Sebag, "Detecting outliers from large data sets," *International Conference on Database Theory*, pp. 225–238, 2009.

[12] T. Rütermann, A. Benabbas, and D. Nicklas, "Know thy quality: Assessment of device detection by wifi signals," in *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 2019, pp. 639–644.

[13] M. Ahmed and M. Denecke, "A survey of adaptive streaming algorithms for time series data," *Data Science and Engineering*, vol. 1, no. 2, pp. 77–88, 2016.

[14] A. M. Townsend, *Smart cities: Big data, civic hackers, and the quest for a new utopia*. WW Norton & Company, 2013.

[15] Y. Lu, F. Wang, and R. Maciejewski, "Building resilience into smart infrastructures," in *2012 6th International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2012, pp. 1–6.

[16] P. Neirotti, A. De Marco, A. C. Cagliano, G. Mangano, and F. Scorrano, "Current trends in smart city initiatives: Some stylised facts," *Cities*, vol. 38, pp. 25–36, 2014.

[17] T. Nam and T. A. Pardo, "Smart city as urban innovation: Focusing on management, policy, and context," *2011 5th International Conference on Theory and Practice of Electronic Governance*, pp. 185–194, 2011.

[18] A. A. Cook, G. Mısırlı, and Z. Fan, "Anomaly detection for iot time-series data: A survey," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6481–6494, 2020.

[19] M. Hoh, A. Schöttl, H. Schaub, and F. Wenninger, "A generative model for anomaly detection in time series data," *Procedia Computer Science*, vol. 200, pp. 629–637, 2022.

[20] K. Choi, J. Yi, C. Park, and S. Yoon, "Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines," *IEEE Access*, vol. 9, pp. 120 043–120 065, 2021.

[21] M. Hu, Z. Ji, K. Yan, Y. Guo, X. Feng, J. Gong, X. Zhao, and L. Dong, "Detecting anomalies in time series data via a meta-feature based approach," *IEEE Access*, vol. 6, pp. 27 760–27 776, 2018.

[22] R. J. Hyndman, E. Wang, and N. Laptev, "Large-scale unusual time series detection," in *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, 2015, pp. 1616–1619.

[23] A. Bustamante, N. Ghimire, P. Sanghavi, A. Pokharel, and V. Irekponor, "Advantages of machine learning in networking-monitoring systems to size network appliances and identify incongruences in data networks," in *Trends in Artificial Intelligence and Computer Engineering. ICAETT 2021. Lecture Notes in Networks and Systems*, vol. 407. Cham: Springer, 2022.

[24] A. Geiger, D. Liu, S. Alnegheimish, A. Cuesta-Infante, and K. Veeramachaneni, "Tadgan: Time series anomaly detection using generative adversarial networks," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 33–43.

[25] M. A. Bashar and R. Nayak, "Tanogan: Time series anomaly detection with generative adversarial networks," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020, pp. 1778–1785.

[26] J. Li, H. Izakian, W. Pedrycz, and I. Jamal, "Clustering-based anomaly detection in multivariate time series data," *Applied Soft Computing*, vol. 100, p. 106919, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494620308577

[27] S. Devlekar, V. Rallapalli, and S. Oswal, "Anomaly detection in time series using unsupervised machine learning approach," *International journal of computer applications*, vol. 184, no. 35, pp. 7–13, 2022.

[28] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[29] G. Strang, *Linear Algebra and Its Applications*, 4th ed. Cengage Learning, 2005.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[31] J. Montiel, M. Halford, A. Dieuleveut, H. Boumaiza, G. Bolmier, R. Sourty, A. Zouitine, A. Atienza, R. Yurchak, Şahin Mersin, V. M. Zabalza, M. Bagdouri, and C. Beckham, "river: Machine learning for streaming data in Python," https://github.com/online-ml/river, 2021.

[32] N. R. Draper and H. Smith, *Applied Regression Analysis*, 3rd ed. Wiley, 1998.

[33] R. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[34] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. Springer, 2013.

[35] C. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate research*, vol. 30, no. 1, pp. 79–82, 2005.