Imposters Among Us: A Supervised Learning Approach to Anomaly Detection in IoT Sensor Data

Tapadhir Das*, Raj Mani Shukla† and Shamik Sengupta*
*Department of Computer Science and Engineering, University of Nevada, Reno, USA
†Department of Computer Science, University of Bristol, UK
Email: tapadhird@nevada.unr.edu, raj.shukla@bristol.ac.uk, ssengupta@unr.edu

***Abstract*—Internet of Things (IoT) technology has made smart homes more prevalent in everyday lives. However, anomalies in IoT data may be emblematic of potential cybersecurity risks like false data injection attacks or physical security risks like house fires. In this paper, we propose a segmentation based anomaly detection method that converts unsupervised time-series data into a supervised format, and then trains a Long-Short Term Memory (LSTM) neural network to detect anomalies. The LSTM network is trained to predict un-sound statistical properties, which gets combined with sound statistical properties, to detect anomalies in IoT sensor data. Data smoothing using Holt Winters Exponential Smoothing is also performed, without loss of information, to improve anomaly detector performance. Using Precision, Recall, and F-Measure scores as metrics, results show efficient anomaly detection performance on IoT temperature sensor data. We, additionally, test performance by varying specific parameters. Lastly, results also show that performing data smoothing, to a certain extent, can improve anomaly detection performance over data that didn't undergo any smoothing.**

***Index Terms*—Anomaly detection, Internet of Things, Machine Learning, Robust Statistics**

I. Introduction

With the increased adoption of Internet of Things technology (IoT), smart homes have become common in our lives. It is estimated that by 2025, there will be 481.9 million smart homes globally [1]. Smart home technology has helped to make everyday lives smarter, connected, efficient, with the goal of increased security .

However, smart home technology also comes with security risks. One risk is cybersecurity attacks which can compromise user privacy and security. Examples of recent cyberattacks on smart homes can be seen in [2], [3], [4]. A prominent cyberattack performed on IoT devices is the false data injection, where criminals tend to strategically and maliciously modify sensor data to corrupt the working functionality of the device. These attacks tend to go unnoticed, which violate user security and privacy [5]. Another security risk is the lack of adequate environmental monitoring for abnormal sensor behavior. Examples include sudden high temperature and carbon monoxide levels in the home, which can be indicative of house fires, or the lights turning on, while the owner is out, which can be suggestive of a burglary. Proper detection of these anomalous behavior patterns can protect the smart home from harm.

This research is supported by the National Science Foundation (NSF) Award #2019164.

Anomaly detection is a continuing research area that is being studied to provide protection against anomalous sensor activity in IoT domains like smart cities [6], electric vehicles [7], IoT network traffic analysis [8] etc. As most IoT devices are commercial off-the-shelf, they lack adequate security monitoring capabilities. Also, due to their inexpensiveness and commercial allurement, security is not given much priority [9]. Therefore, IoT devices and smart homes need to be protected from potentially damaging anomalous behavior.

Another noteworthy problem with IoT sensors is the subjection to sensor noise, which can affect performance. Noise may be introduced by the actual measurement of the sensor, or even from random variables during data gathering. This can degrade anomaly detection performance. Hence, it is critical to come up with anomaly detection approaches that process out noise without compromising performance.

In this paper, we propose an anomaly detection method using a Long-Short Term Memory (LSTM) neural network. An LSTM is employed as it is a standard architecture that is used to conduct machine learning analysis for time-series data, due to its capability of identifying patterns over long sequences. We use an open source IoT sensor dataset, consisting of unsupervised sensor data. Our approach first performs data smoothing on this data to remove inherent noise from the dataset. Following this, we create a supervised dataset from the data, that is used to train the model, which is used for anomaly detection. Our main contributions include:

- Performing data smoothing on our dataset to remove sensor noise.
- Converting unsupervised time-series data into a supervised format for LSTM training.
- Modeling normal sensor behavior using an LSTM network, using robust statistical properties.
- Mathematically modeling anomalies to check model efficacy at anomaly detection.

The remainder of the paper is structured as follows: Section II provides a history of the related research that has been conducted in anomaly detection within the IoT domain. The research scenario and our proposed anomaly detection approach is shown in Section III. Our experimentation, results, and analysis are provided in Section IV. Finally, conclusions are drawn in Section V.

II. Related Works

Robust anomaly detection methods are continually being proposed for improved performance in IoT. [9] attempted to detect anomalies in a smart home, using a hidden Markov model (HMM). The method gave promising results, however, HMMs are prone to not detecting uncommon anomalies efficiently [10]. Other researchers have employed statistical approaches [11] [12]. In [11], researchers conducted network traffic anomaly detection using Principal Component Analysis (PCA), while in [12], network traffic anomaly detection was performed using wavelet analysis. However, statistical approaches are limited as their selected thresholds are usually not representative of real world threat scenarios. Also, statistical approaches like PCA and wavelet analysis can't be used on time-series data.

Clustering algorithms also tend to be viable approaches [13] [14]. In [13], the authors proposed CLAPP, which is a self-constructing feature clustering approach for anomaly detection, and they used this to conduct intrusion detection. The work conducted in [14] proposed using a fuzzy clustering based artificial neural network (ANN) for intrusion detection in cloud computing environments. Clustering methods can be used to detect anomalies in time-series data. However, these approaches depend on the normal state of the system to make accurate predictions. IoT sensor data, on the other hand, may change their normal state over time, which may lead to misclassifications. Also, this could lead to clustering algorithms getting trapped at a local minima [10].

Another prominent technique for time-series anomaly detection in IoT is deep learning [6], [15], [16]. In [6], the authors proposed a convolutional neural network (CNN) based anomaly detector, capable of detecting point, contextual, and discord anomalies in time-series data. The limitation with this approach is that CNNs can't account for the historical value of a time-series data point, which may be important for anomaly detection and forecasting. The researchers in [15] proposed an anomaly detection method by using an LSTM neural network, where the model predicts anomalies depending on error computation. The authors in [16] propose an supervised LSTM anomaly detection method where they predict anomalies based on statistics from the dataset. The limitation on both the above approaches is that these methods are susceptible to noisy data. Noise from sensors are a common additive feature to sensor values, which make misclassfications a high probability.

In contrast to the previously studied techniques, our work proposes a supervised LSTM, that uses dataset statistics, to identify anomalies. This is significant as there are limited studies conducted for supervised anomaly detection in time-series data. The proposed method also performs data smoothing prior to training, in order to minimize the impact of data noise on anomaly detection performance, which hasn't been investigated.

III. System Model and Methodology

The dataset consists of time-series data from an indoor environment [17]. The sensor being polled is a temperature sensor from a single location. The sensor gets polled every 31 seconds. We are considering the temperature readings from February 28th to March 21st, 2004. We use a time unit *i* as a discrete value and a natural number, where *i* represents a particular time slot. The temperature sensor readings are represented by *x*. The temperature reading at a particular time unit *i* is represented as x_i. The goal is to determine if these temperature readings x_i are anomalous or normal. The final sensor reading time is denoted as *E*.

The proposed anomaly detection architecture is illustrated in Fig. 1. The main steps of the proposed approach is highlighted in the following subsections:

A. Data Smoothing

To ensure that our data is minimally affected by noise, we perform Holt-Winters Exponential Smoothing [18]. This method was chosen as it is an eminent method to perform smoothing on data containing seasonality and trend, which the target dataset carries. Data smoothing methods that don't take seasonality and trend into consideration are avoided as they may reduce performance. There are three operating parameters: the *data smoothing factor* denoted by α, where $0 \leq \alpha \leq 1$, the *trend smoothing factor* denoted by β, where $0 \leq \beta \leq 1$, and the *seasonal change smoothing factor* denoted by γ, where $0 \leq \gamma \leq 1$. The magnitudes of α, β and γ, are inversely proportional to the amount of smoothing performed. These smoothing constants determine how quickly the weights of the series decay for the current observation. Values closer to 1 weigh recent observations heavily, while values closer to 0 give weight to past observations [19]. α is the primary variable for data smoothing, as it establishes the most influence on the level of smoothing. $\alpha = 1$ means that the dataset has not undergone any smoothing, while $\alpha = 0$ indicates maximal smoothing. The seasonal period of the time-series data is symbolized by ρ. The *smoothed data level* L_i at time *i* is given by:

$$L_i = \alpha(x_i - S_{i-\rho}) + 1 - \alpha(L_{i-1} + T_{i-1}), i = \{0, 1, ...E\} \quad (1)$$

The *trend* T_i of the data at time *i* is illustrated with:

$$T_i = \beta(L_i - L_{i-1}) + (1 - \beta)T_{i-1}, i = \{0, 1, ...E\} \quad (2)$$

The trend T_i represents the slope of the data trend at time *i* [19]. Correspondingly, the *seasonal component* S_i of the data at time *i* is provided by:

$$S_i = \gamma(x_i - L_i) + (1 - \gamma)S_{i-\rho}, i = \{0, 1, ...E\} \quad (3)$$

The *seasonal component* S_i symbolizes a weighted average between the current seasonal index, and the seasonal index of the same time during the last season [20]. Finally, the *forecasted time-series value* $\hat{x_i}$ for the data at time *i* can be computed using:

$$\hat{x_i} = L_{i-1} + T_{i-1} + S_{i-\rho}, i = \{0, 1, ...E\} \quad (4)$$

Using this technique, the time-series data becomes more representative of the normal state of the sensor, with minimal interference from noise.

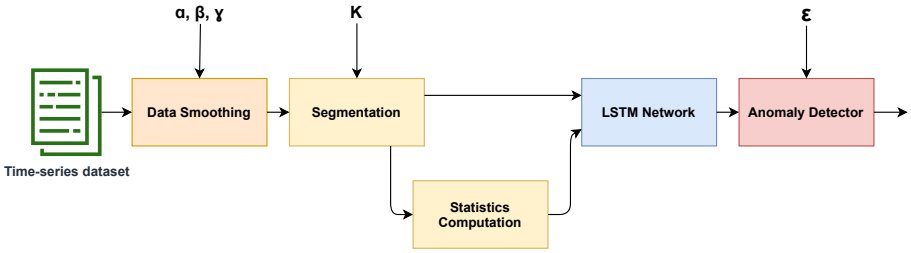

Fig. 1: Anomaly Detection Architecture

1) Finding optimal data smoothing factor: Data smoothing should be performed optimally, so minimal information is lost from the original data. Simultaneously, data smoothing should also minimize the noise. This makes it important to select the data smoothing factor α that ensures maximum performance, while minimizing the information loss from smoothing. The appropriate α can be selected by abiding to the inequality:

$$M - M' < \lambda \tag{5}$$

where M is the mean of the standard deviations of all the non-smooth data segments. Dataset segmentation is explained in the following subsection. M can be computed by:

$$M = \frac{\sum_{j=0}^{N} \sqrt{\frac{1}{K}\sum_{i=0}^{K-1}(x_i - \bar{x_i})^2}}{N} \tag{6}$$

where $\bar{x_i}$ represents the mean of the sensor values in that segment. Correspondingly, M' is the mean of the standard deviations of all the smooth data segments smoothed by α, and is computed by:

$$M' = \frac{\sum_{j=0}^{N} \sqrt{\frac{1}{K}\sum_{i=0}^{K-1}(\hat{x}_i - \bar{\hat{x}_i})^2}}{N} \tag{7}$$

where $\bar{\hat{x}_i}$ represents the mean of the sensor values in that segment. Finally, we introduce λ which is the *strength of anomaly*, where $\lambda \geq 0$. The magnitude of λ will further sway the value of the anomalies from the expected sensor value, meaning that a lower λ value looks more like normal sensor data, and is more difficult to detect, than a higher λ value. Optimal α is selected when the value of λ is minimized in 5.

B. Segmentation

Post data smoothing comes the data segmentation module. In this module, we split the smoothed data into segments of size *K*. The goal behind segmentation is to effectively identify anomalies in a localized context. Each segment is represented as Seg_i, where $Seg_i = \{x_i, x_{i+1}, ...x_{K-1}\}$ and *i* represents the starting time slot for that segment. We also assume the total number of segments to be *N*. Every segment gets sorted in ascending order of their values. Then, the middle 50% of the segment is extracted. We assume that the middle 50% contains no anomalies, as they are representative of normal sensor data. The other 25% on either side may or may not contain any anomalies. We represent the middle 50% values of all segments as $Seg_{i,m}$.

C. Statistics Computation

For every single segment Seg_i, we must also compute statistics for that particular segment. In our case, we compute the M-estimator for every Seg_i. M-estimator tends to be robust when there are anomalies in a dataset, as they use median in their construction. Median, in comparison to mean, is not easily swayed by anomalies. This makes them a better fit for anomaly detection. The M-estimator for a particular segment can be computed using:

$$\sum_{i=0}^{K-1} \eta(\frac{x_i - \mu_j}{\sigma(Seg_i)}) = 0, j = \{0, 1, ...N\} \tag{8}$$

where $\sigma(Seg_i)$ represents a function on Seg_i which provides the initial estimate that may be mean or median. The variable μ_j, the solution of the equation, represents the M-estimator of the segment Seg_i. Lastly, η represents a real value Huber function which is denoted by:

$$\eta(x) = x.min(1, \frac{b}{|x|}) \tag{9}$$

where b is a constant value. The computed statistics are essential for anomaly detection further down in this process.

D. Deviation computation

We aim to convert our unsupervised dataset into a supervised one for training. After we compute the statistics, we separate the segments into training and testing segments. The goal is to compute the acceptable deviations d_j of each training segment from the μ_j of that segment. The formula to compute deviations for the training segments is given by:

$$d_j = max((|\mu_j - min(Seg_i)|), (|\mu_j - max(Seg_i)|)), \\ i = \{0, 1, ...E\}, j = \{0, 1, ...N\} \tag{10}$$

However, we don't compute the deviations for the testing segments. The reason is because the testing segments are the ones that will contain anomalies, which may negatively influence direct deviation computation. Hence, we plan to predict the deviations for each testing segment, from the middle 50% of the segments.

E. LSTM Network

We are using an LSTM network for training. This network will be used to predict the deviations for the testing segments. The inputs to the network are $Seg_{i,m}$, and the corresponding labels are d_j. Fig. 2 illustrates the proposed LSTM.

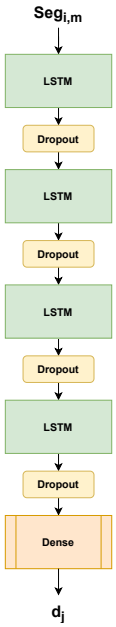

Fig. 2: LSTM Network

F. Anomaly Detector

The anomaly detector is responsible for detecting anomalies in the testing segments. An essential component of the anomaly detection approach is the ϵ, where $\epsilon \geq 0$. Another important component of the anomaly detection approach is the *training segment threshold T*. These parameters allow additional control over the anomaly detector to ensure proper anomaly identification. T is computed only on the training segments, by the following equation:

$$T = \frac{\sum_{j=0}^{N} \sqrt{\frac{1}{K} \sum_{i=0}^{K-1} (\hat{x}_i - \bar{\hat{x}}_i)^2}}{N} \tag{11}$$

where $\bar{\hat{x}}_i$ represents the mean of the sensor values in that training segment. Once the deviations of the testing data are predicted, we combine the deviations and the previously obtained statistical properties to check if a sensor value is an anomaly or normal. The anomaly test is presented as:

$$\hat{x}_i = \begin{cases} 1 & \text{if}(\hat{x}_i > \mu_j + \epsilon T d_j), \\ 1 & \text{if}(\hat{x}_i < \mu_j - \epsilon T d_j), \\ 0 & \text{otherwise}, \end{cases} \tag{12}$$

The above equation is computed to test if a sensor value $\hat{x}_i$, in the testing segment, is equal to 1 (anomaly) or 0 (normal).

IV. Simulation and Results

Our approach was implemented in python, using the tensorflow library. The dataset used was the Intel Berkeley Research Lab Dataset [17], specifically the temperature sensor data from the first sensor node. Once the network was trained, we embedded anomalies in the testing data. Then, we performed experiments to measure the efficiency of our approach.

TABLE I: Performance against positive anomalies

λ	P	R	F
0	5.93%	39.13%	10.30%
2	15.67%	75.41%	25.95%
4	21.64%	99.59%	35.55%
6	78.39%	99.59%	87.73%
8	97.60%	100.00%	98.79%
10	97.60%	100.00%	98.79%
12	97.60%	100.00%	98.79%
14	97.60%	100.00%	98.79%
16	97.60%	100.00%	98.79%
18	98.39%	100.00%	99.19%

TABLE II: Performance against negative anomalies

λ	P	R	F
0	6.71%	45.90%	11.71%
2	11.44%	82.38%	20.09%
4	13.65%	99.59%	24.01%
6	13.65%	99.59%	24.01%
8	14.30%	99.59%	25.01%
10	17.08%	99.59%	29.15%
12	24.16%	99.59%	38.88%
14	34.86%	100.00%	51.69%
16	45.61%	100.00%	62.64%
18	56.22%	100.00%	71.98%

A. Anomaly Generation

To embed anomalies in the testing data, we manipulated measured sensor values. This manipulation can symbolize a false data injection or even a physical scenario like a house fire. Our manipulation included both positively and negatively scaled anomalies, which are referred to as positive and negative anomalies, respectively, from here on. The positive anomalies have values more than the expected sensor value. Correspondingly, the negative anomalies have values less than the expected sensor value. Positive anomalies are therefore embedded using:

$$\hat{x}_i = \hat{x}_i + \lambda T \tag{13}$$

while negative anomalies are denoted by:

$$\hat{x}_i = \hat{x}_i - \lambda T \tag{14}$$

The positive and negative anomalies are added to the upper and lower 25% of the segments respectively.

B. Experimentation

In our experiments, we set up the following initial values: $K = 16, \alpha = 0.3, \beta = 0.05, \gamma = 0.05, \lambda = 6, \epsilon = 5$. As our dataset hasn't undergone any fundamental change in values, and consist of mostly steady data with random noisy fluctuations, the value of α, β, and γ should be on the lower end of the spectrum [19]. The values of λ and ϵ are not strict and must be chosen according to the statistics of the dataset being used: standard deviation, maximum and minimum values, and range. For performance metric computation, we are using Precision *P*, Recall *R*, and F-Measure *F*.

We experiment with our approach to see how effectively it can detect positive and negative anomalies. Table I shows performance against positive anomalies with varying λ =

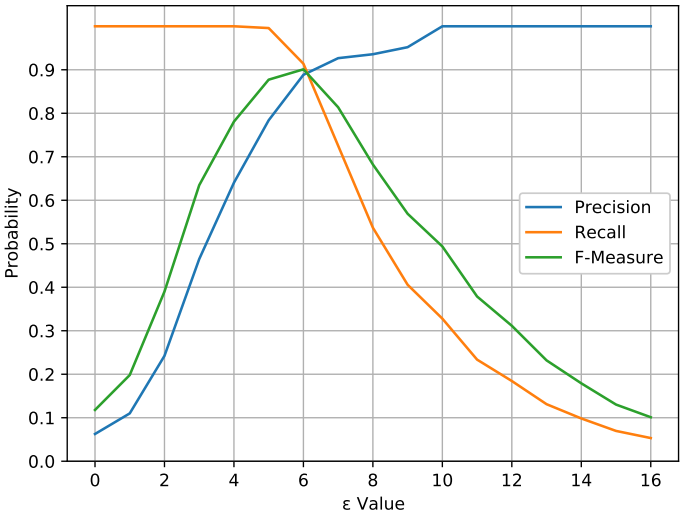

Fig. 3: P, R, F against varying ϵ

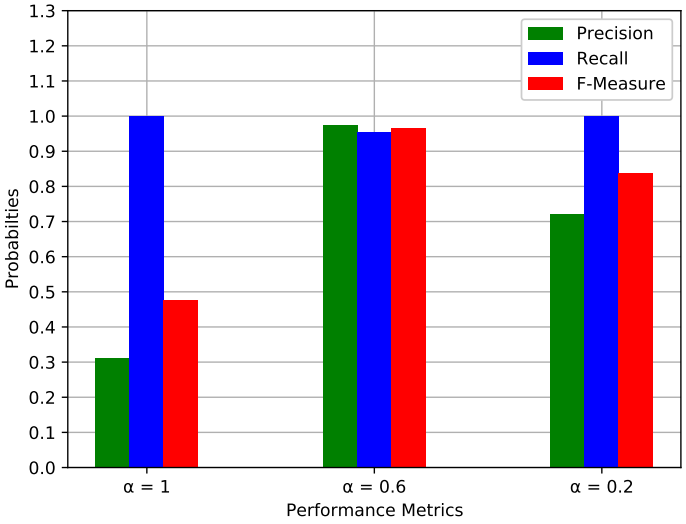

Fig. 4: P, R, F against varying α

$\{0, 2, 4...18\}$. We see that the values of P, R, and F increase as λ increases. This is expected as larger anomalies are more evident and easier to detect in context of a segment. Table II provides the performance against negative anomalies with varying λ. From this table, it also shows that P, R, and F scores increase as the value of λ increase. The approach, comparatively, slightly under-performs in detecting negative anomalies, as the dataset is solely filled with positive values and the value of T is low, compared to the range of the dataset. Hence, it takes a higher magnitude of λ to show effective performance in detecting negative anomalies.

Subsequently, we showcase the performance of the approach by varying the parameter $\epsilon = \{0, 1, ...16\}$. Fig. 3 illustrates the performance of the anomaly detection method as the value of ϵ is varied. We can see changes in the performance metrics when the parameter ϵ is gradually increased. An inverse correlation is noted between the observed P and R. The value of R is higher at low values of ϵ, while the value of P is higher at high values of ϵ. We also note that the rate of increase in P and the rate of decrease in R appears to be approximately the same. The best balance between P and R is observed at $\epsilon = 6$, where the $F = 0.9$.

Next we analyze the effect of data smoothing on the performance of the anomaly detection approach, by varying $\alpha = \{1, 0.6, 0.2\}$. Fig. 4 illustrates the P, R, and F values, when the data has undergone no smoothing ($\alpha = 1$), moderate smoothing ($\alpha = 0.6$), and intense smoothing ($\alpha = 0.2$). We see that R values are nearly similar across all cases. However, both the P and F scores are lowest when there is no data smoothing performed. In comparison, P and F scores are higher in the scenario where the data has undergone intense smoothing. The highest P and F scores are recorded when the dataset has passed through moderate smoothing. From this we can see that data smoothing, to a moderate extent, increases anomaly detection efficiency. Intense smoothing or higher can lead to a decrease in performance.

Finally, we analyze the impact of the data-smoothing factor

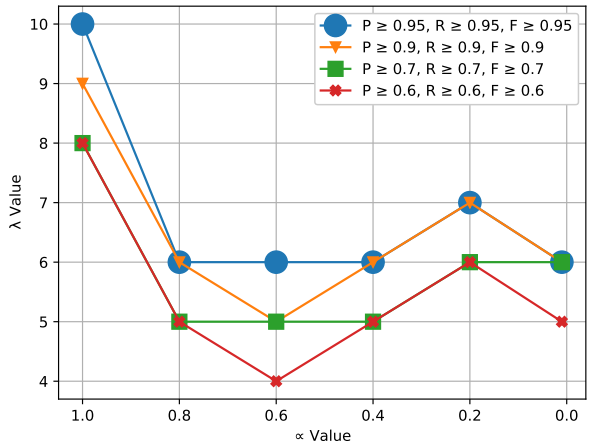

Fig. 5: Effect of data smoothing factor α on anomaly strength λ under varied P, R, F conditions

α to the detectability of the anomaly strength λ. This is studied by varying the value of $\alpha = \{1.0, 0.8, 0.6, 0.4, 0.2, 0.01\}$. We assume that the detection approach, trained on a dataset that has been smoothed with α, can efficiently detect anomalies of a certain strength λ, if it achieves a certain baseline performance for P, R, and F. We examine four scenarios:

1) $P \geq 0.6$, $R \geq 0.6$, $F \geq 0.6$.
2) $P \geq 0.7$, $R \geq 0.7$, $F \geq 0.7$.
3) $P \geq 0.9$, $R \geq 0.9$, $F \geq 0.9$.
4) $P \geq 0.95$, $R \geq 0.95$, $F \geq 0.95$.

the Fig. 5 shows the impact of α and the kind of anomaly strength λ it can efficiently detect, as we increase the expectation for the baseline performance of P, R, and F. From the figure, we observe that the anomaly detector exhibits similar behavior across all four scenarios. Values of λ increase, for every α, as we go from scenario 1 to 4. This is to be expected, as we are increasing the expectation of baseline performance for P, R, and F. In scenarios 1,2,3, and 4, we notice that when

there is no data smoothing performed ($\alpha = 1$), the anomaly detector can effectively detect higher λ anomalies of 8,8,9, and 10, respectively.

The performance of the anomaly detector improves between $0.8 \leq \alpha \leq 0.4$, in all four situations. The best anomaly detection performance in scenario 1 and 3 was $\alpha = 0.6$ as it yielded the least λ values of 4 and 5, respectively. In scenario 2 and 4, the best performance was observed when $0.8 \leq \alpha \leq 0.4$ as it efficiently detected anomalies of λ value 5 and 6, respectively. However as data smoothing continued to increase, we noticed that the performance of the anomaly detector decreased across all scenarios. In scenarios 1,2,3, and 4, a dataset smoothed with $\alpha = 0.2$ yields λ values of 6,6,7, and 7, respectively. Correspondingly, $\alpha = 0.01$ effectively detected anomalies of λ values of 5,6,6, and 6, respectively.

Therefore, we can demonstrate that non smoothed data will only help the anomaly detector effectively detect anomalies of higher λ, but do not perform optimally when there are harder to detect anomalies. We observe that data smoothing helps increase anomaly detection efficiency, specifically with harder to detect anomalies. This efficiency is best when the data is smoothed with $0.8 \leq \alpha \leq 0.4$. In certain scenarios, this efficiency is shown to be maximum when $\alpha = 0.6$. However, excessive data smoothing ($\alpha < 0.4$) is detrimental to performance as it eliminates many essential data points, which lead to incorrect forecasting. Also, the presence of anomalies in the dataset disrupt performance on excessively smoothed data. The results and analysis presented are not a generalized solution, and the location of optimal data smoothing might change depending on the dataset being used. Optimal α should be computed using 5.

V. Conclusion

In this paper, we proposed an anomaly detection approach that was applied to IoT time-series data. This technique converted unsupervised data to a supervised format, for training. Our approach included a segmentation based method to effectively detect anomalies in a localized context. Data smoothing was also performed to increase anomaly detection efficacy. We measured performance from various perspectives: regulating parameters, anomaly strengths, while using *P*, *R*, and *F*. Results showed that the proposed approach performed better as anomaly strength λ increased for both positive and negative anomalies. It also showed that *P* and *R* inversely correlated to one another as ϵ increased, and $\epsilon = 6.0$ yielded the best *F* score. Results also highlighted that the best *P*, *R*, and *F* scores were achieved by data that had been moderately smoothed ($0.8 \leq \alpha \leq 0.4$), compared to intensely smoothed or non-smoothed data. This provided better anomaly detection performance, specifically on harder to detect anomalies, over a dataset that had undergone excessive smoothing or none at all.

References

[1] J. Lasquety-Reyes, "Smart Home - number of households in the segment Smart Home worldwide 2025." [Online]. Available: https://www.statista.com/forecasts/887613/number-of-smart-homes-in-the-smart-home-market-worldwide

[2] S. Cangeloso, "Philips Hue LED smart lights hacked, home blacked out by security researcher - ExtremeTech." [Online]. Available: https://www.extremetech.com/electronics/163972-philips-hue-led-smart-lights-hacked-whole-homes-blacked-out-by-security-researcher

[3] C. Domonoske, "S.C. Mom Says Baby Monitor Was Hacked; Experts Say Many Devices Are Vulnerable." [Online]. Available: https://www.npr.org/sections/thetwo-way/2018/06/05/617196788/s-c-mom-says-baby-monitor-was-hacked-experts-say-many-devices-are-vulnerable

[4] J. Maher, "Couple say hacker took over their smart home, playing vulgar music and setting thermostat to 90 degrees," Sep. 2019. [Online]. Available: https://www.newsweek.com/google-nest-hack-milwaukee-1460806

[5] M. Ahmed and A.-S. K. Pathan, "False data injection attack (fdia): an overview and new metrics for fair evaluation of its countermeasure," *Complex Adaptive Systems Modeling*, vol. 8, pp. 1–14, 2020.

[6] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, and H. Ming, "Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2019, pp. 0305–0310.

[7] O. Avatefipour, A. S. Al-Sumaiti, A. M. El-Sherbeeny, E. M. Awwad, M. A. Elmeligy, M. A. Mohamed, and H. Malik, "An intelligent secured framework for cyberattack detection in electric vehicles' can bus using machine learning," *IEEE Access*, vol. 7, pp. 127 580–127 592, 2019.

[8] D. H. Hoang and H. D. Nguyen, "A pca-based method for iot network traffic anomaly detection," in *2018 20th International conference on advanced communication technology (ICACT)*. IEEE, 2018, pp. 381–386.

[9] S. Ramapatruni, S. N. Narayanan, S. Mittal, A. Joshi, and K. Joshi, "Anomaly detection models for smart home security," in *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*. IEEE, 2019, pp. 19–24.

[10] G. Fernandes, J. J. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença, "A comprehensive survey on network anomaly detection," *Telecommunication Systems*, vol. 70, no. 3, pp. 447–489, 2019.

[11] F. Harrou, F. Kadri, S. Chaabane, C. Tahon, and Y. Sun, "Improved principal component analysis for anomaly detection: Application to an emergency department," *Computers & Industrial Engineering*, vol. 88, pp. 63–77, 2015.

[12] Z. Du, L. Ma, H. Li, Q. Li, G. Sun, and Z. Liu, "Network traffic anomaly detection based on wavelet analysis," in *2018 IEEE 16th International Conference on Software Engineering Research, Management and Applications (SERA)*. IEEE, 2018, pp. 94–101.

[13] R. K. Gunupudi, M. Nimmala, N. Gugulothu, and S. R. Gali, "Clapp: A self constructing feature clustering approach for anomaly detection," *Future Generation Computer Systems*, vol. 74, pp. 417–429, 2017.

[14] N. Pandeeswari and G. Kumar, "Anomaly detection system in cloud environment using fuzzy clustering based ann," *Mobile Networks and Applications*, vol. 21, no. 3, pp. 494–505, 2016.

[15] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings*, vol. 89. Presses universitaires de Louvain, 2015, pp. 89–94.

[16] W. Jia, R. M. Shukla, and S. Sengupta, "Anomaly detection using supervised learning and multiple statistical methods," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, 2019, pp. 1291–1297.

[17] P. Bodik, W. Hong, C. Guestrin, S. Madden, M. Paskin, and R. Thibaux, "Intel lab data," *Online dataset*, 2004.

[18] P. R. Winters, "Forecasting sales by exponentially weighted moving averages," *Management science*, vol. 6, no. 3, pp. 324–342, 1960.

[19] N. S. Software, "Exponential smoothing – trend seasonal," https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Exponential_Smoothing-Trend_and_Seasonal.pdf.

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