Anomaly Detection using Supervised Learning and Multiple Statistical Methods

Watson Jia*
Princeton University
Email: *watsonj@princeton.edu

Raj Mani Shukla†
University of Nevada, Reno
Email: †rshukla@unr.edu

Shamik Sengupta‡
University of Nevada, Reno
Email: ‡ssengupta@unr.edu

***Abstract*—The presence of anomalies or outliers within time-series data can have a detrimental effect on the efficiency of automated decision-making applications. For example, in the context of vehicular traffic flow, various services reliant on traffic data may be negatively impacted by anomalies. This paper presents an automated anomaly detection method based on supervised Long-Short Term Memory (LSTM) neural network and statistical analysis. We train LSTM neural network to predict non-robust statistical properties and combine them with robust properties to determine the anomalies in time-series data. The proposed method relies on segmentation and tunable parameters for anomaly test. We measure the efficacy of our method in terms of Precision, Recall, and F-measure. The metrics approach to 100% for certain instances. We also analyzed the performance on the prevalence of anomalies and on varying specific parameters of the model.**

***Index Terms*—Anomaly detection, Robust statistics, Time-series, Internet of Things**

I. Introduction

The expansion of the Internet of Things (IoT) has enabled the society that has become much more integrated with technology than in the past. Sensors have played an integral role in enabling the growth of IoT and have enabled numerous technological applications in various fields. Through an Internet connection, remotely controlled sensors can gather large amounts of valuable information from the environment and organize the collected data in a central location for analysis [1]. One application is in traffic flow, where sensors can collect traffic information from roads in the form of a time-series. By taking advantage of patterns and periodic behavior in these time-series, there are many potential applications, such as helping provide for better traffic management by authorities or travel decisions by a commuter [2]. Sensors have many other applications such as in smart homes, electric vehicles, or enabling smart grids, and the effectiveness of these applications can have many benefits for individuals as well as society at large [3] [4] .

However, the data collected by sensors is non-deterministic and can be affected by outside factors. This leads to the presence of anomalies or outliers within these time-series data, which can have an adverse effect on the effectiveness of automated decision-making IoT applications, especially if they are reliant on statistical methods [5]. A variety of factors can cause anomalies. One source is external environmental factors; for example, in the case of traffic flow, a car accident could lead to an unusual dip in traffic detected by sensors at a given location. In time-series data, this may be manifested as a sudden drop in traffic flow for a location. Another source can be the quality of the sensor itself; if the sensor is not correctly calibrated or physically tampered with, a given sensor will not be able to collect accurate information. Anomalies can be malicious; in a data falsification attack, a malicious actor targets a sensor or a group of sensors and intentionally modifies the data values collected [6]. This is an area of significant concern in the IoT since malicious anomalies can negatively impact the effectiveness of decision-making capabilities in IoT-enabled applications [7]. Anomalies negatively impact automated statistical analysis in IoT applications. For example, taxi service may be negatively impacted if it is using data that has been corrupted or otherwise of bad quality. Thus, it is of great interest that we be able to detect anomalies within time-series data to ensure the effectiveness of data analysis applications. We explore the problem of anomaly detection in time-series data from the perspective of traffic flow data collected from sensors throughout the state of California.

The problem of anomaly detection presents several challenges. For one, there are myriad of sensors connected to the IoT, each of which can collect a wide variety of data. Moreover, in the context of our problem, different traffic flow sensors have different challenges that must be addressed. For example, some sensors are located remotely and experience adverse conditions in comparison to sensors located in urban areas. Due to the sheer quantity of data that must be considered as well as challenges with each sensor, manual anomaly detection is infeasible. Automated statistical analysis must be conducted instead to be able to process these large amounts of data quickly and identify anomalies within them. This suggests a supervised learning approach; however, there is a lack of labeled anomaly data to be able to train learning models for this problem. Moreover, there is the question of what statistical methods and tests may be effective in anomaly detection that is, what statistical metrics may be robust to anomaly influences so that they can identify anomalies with a high degree of accuracy.

There are multiple statistical properties that time-series data can exhibit, such as mean, median, and M-estimator . These properties are often used in statistical anomaly detection tests, each with their own advantages and disadvantages [8]. The mean can be accurate in describing the central tendencies of

978-1-7281-4550-1/19/$31.00 ©2019 IEEE
DOI 10.1109/ICMLA.2019.00211

normally distributed data-sets as well as skewed distributions such as log-normal and Poisson distributions. However, the mean is not a robust statistic and is heavily influenced by the presence of anomalies. In contrast, the median is a robust measure of central tendency but may not be as accurate in describing the central tendencies of skewed distributions. M-estimators have the benefits of being robust as well as being able to provide a sample average. However, computing M-estimators is more complex and involved. In this paper, we combine these three statistical properties of the data samples to detect anomalies. Furthermore, once anomalies are present in data, the robust statistical measures may not deviate, but the minimum and maximum values of the data-set may change, thus affecting the maximum deviation of the data. We employ a Long-Short Term Memory (LSTM) based neural network to predict deviations in segments of our time-series data as if anomalies were not present in our data. More importantly, an LSTM model is well suited to apply predictions to time-series data, since it can take advantage of longer-term patterns within the data as well as not being affected by large gaps between important events in the time-series. We then combine statistical metrics with supervised learning to be able to address the anomaly detection problem.

This paper attempts to address the anomaly detection problem in the context of traffic flow data in the form of a time-series collected from IoT sensors. The main contributions of this work are as follows.

- We employ LSTM neural network to estimate non-robust statistical property and combine it with robust statistical property of a sample.
- We combine the statistical analysis with supervised learning, LSTM neural network, to determine the anomalies in time-series segments.
- We also verify the effectiveness of our approach using realistic data traces from California freeways.

The rest of the paper is organized as follows: Section II briefly describes the literature survey. Section III presents the proposed system model. Section IV provides a description of the problem statement and proposed method. Section V describes experimental process and section VI concludes this paper mentioning the direction of future research scopes.

II. Literature Survey

There has been work done in the literature regarding anomaly detection. Supervised learning to train on labeled anomaly has often been used in literature. In [9], Malhotra et al. have proposed an anomaly detection model for time-series data that has used LSTM neural network. The given method predicts time-series value, and based on the error, the anomaly test is performed. However, the given method has used LSTM network like a time-series predictor. Although the method has been used for anomaly detection, this kind of model is found to be more successful when used as predicting future values of a time-series [10], [11]. Further, any concrete statistical analysis has not been done in the given work. In [12], Salman et al. have used Linear Regression and Random Forest for anomaly detection and their categorization in Multi-cloud environment. The method proves to provide very high accuracy (99% detection, 93.6% categorization) for a particular data-set. However, the method has used a labeled data-set where anomalies and their types are known prior to the training model. Most of the time, anomalies are not known in advance, or new kinds of anomalies appear. Supervised learning on labeled anomalies cannot be used efficiently in such cases as enough training data is not available.

Principal Component Analysis (PCA) has also been used for anomaly detection. Netflix has proposed an anomaly detection method using the PCA [13]. Here, the methodology finds any anomaly regarding the unusual sign-in process or any failed banking transaction to alert end users. However, the data that has been used here is in a different form and not time-series data.

Clustering is also another technique used in the literature to detect anomalies. In [14], Li et al. have proposed a method for representing features cluster center and nearest neighbor (CANN). The proposed method performs better than the traditional k-NN or SVM based approach. In [15], Pandeeserai et al. have proposed an anomaly detection method using a hybrid of Fuzzy C-Means clustering algorithm and Artificial Neural Network (FCM-ANN). The presented method outperforms than the Nave Bayes classifier and classic Artificial Neural Network (ANN) even for low-frequency attacks. However, in IoT sensors, new valid patterns tend to occur frequently [16]. Thus, misclassification problem in clustering-based approach may disrupt the primary purpose of anomaly detection.

This paper has used LSTM based supervised learning in combination with the statistical properties of the time-series data to detect anomalies. In contrast to traditional supervised learning-based approaches that have used labeled anomalies for training a model, we train the model to estimate non-robust statistical properties and deviations in data. Thus in our method, the disadvantage due to not-availability of labeled anomalies and occurrence of new patterns in anomalies are avoided.

III. System model and problem statement

In this section, we describe the overview of the proposed system architecture. We also formally describe the anomaly detection problem.

A. System model

The model consists of distributed sensors that collect traffic data in the state of California and send it to a centralized server [17]. The centralization makes IoT-based applications to be easily developed [18]. Each sensor is a physical inductive-loop traffic detector located at a specific location that records certain statistics, including its location, which is denoted in terms of a positive real number $PostMile$, a traffic flow statistic denoted as $AggFlow$, and a traffic speed statistic denoted as $AggSpeed$. Every five minutes, the traffic sensor records an $AggFlow$ and $AggSpeed$ value, and through an internet connection, the sensor sends this data to a centralized

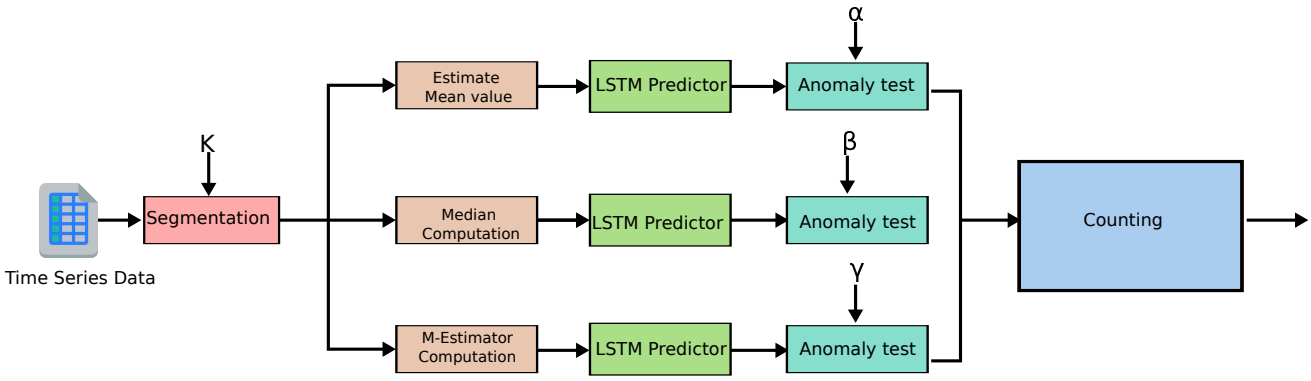

Fig. 1. Anomaly Detection Model

server. We consider the traffic flow statistics collected by all sensors during October 2017. All time-series data collected by the sensors are held in the centralized server where they are viewed and analyzed by transit authorities, academic researchers, and the general public. For the anomaly detection problem, we consider the time-series of traffic flow data at a particular sensor location.

B. Problem Statement

We use discrete-time unit i such that i is a set of natural number, and it represents a particular time-slot. The length of the discrete-time slot is τ. The time-series vehicular traffic count values at a time-slot i is represented using the variable x_i. We divide time-series data into segments of size k such that a segment starting at time-slot i is represented as TS_i and is equal to the vector $\{x_i, x_{i+1}....x_{k-1}\}$. The problem statement is given a segment of values $\{x_i, x_{i+1}....x_{k-1}\}$, determine the points in the segment that are anomaly or outliers. These points are those that deviate significantly from their usual value.

IV. Methodology

Figure 1 shows the overview of the proposed architecture to determine anomalies in time-series data. We analyze time-series data and based on that, determine whether a particular point is anomalous or not. The different steps used in the given anomaly detection problem has been outlined below.

A. Segmentation

First, we split the raw time-series data into segments. The segmentation of time-series is done according to the parameter k, where k is the length of each segment. Segmentation allows us to identify anomalies in their localized contexts. For every such segment, we assume that the middle 50% of values are not anomalies. That is if we sort a segment in ascending order of their values, then the values that lie in the middle 50% portion are assumed to be true values. These middle half values are represented as $TS_{m,i}$. The other half values at extreme ends may or may not be anomalies. The proposed method finds the anomaly test in 50% of the values that are in extreme ends.

B. Statistical analysis

For every segment, we find its statistical properties either using LSTM-based estimator or directly computing it. Thus, for every segment, we first estimate the mean of the segment. That is we find $mean(TS_i)$. Since TS_i itself may have anomalies the actual mean values may not be a real one as even the presence of a single anomaly may changes the mean value of TS_i. Therefore, we employ the LSTM neural network to predict mean value rather than calculating directly from TS_i. The LSTM model that we developed takes $TS_{m,i}$ such that middle half values as input and based on that predicts the mean of the whole segment $mean(TS_i)$. Thus the points that may be anomaly have not been used at all in our LSTM model, and only clean data is used for predicting the mean of the whole segment.

We calculate the median for every time segment directly using vector TS_i. The median lies in the middle portion of the segment. Since we have already assumed that the 50% values that lie between a segment are not anomalies the median for the whole segment can be calculated directly. Thus median is supposed to be robust even if there are 50% anomalies in a segment. This implies that the value of median does not change in the presence of anomalies.

We further calculate M-estimator for every segment TS_i. The critical property of the M-estimator is that it is also resilient to the presence of anomalies. Moreover, it also does not depend on samples having normal distribution. The M-estimator for a sample TS_i is obtained as the solution of the equation 1.

$$\Sigma_{i=i}^{k-1}\xi(\frac{x_i-\mu}{\sigma(TS_i)}) = 0 \tag{1}$$

The denominator $\sigma(TS_i)$ is a function on sample TS_i and gives initial estimate which can be mean or median. The solution of the equation μ is the robust M-estimator for sample TS_i. ξ is a real valued Huber function $\xi(x) = x.min(1, \frac{b}{|x|})$, where b is a constant.

Thus using statistical analysis, we derive three properties of a segment. First, we estimate the mean using LSTM neural network, and then we calculate median and M-estimator

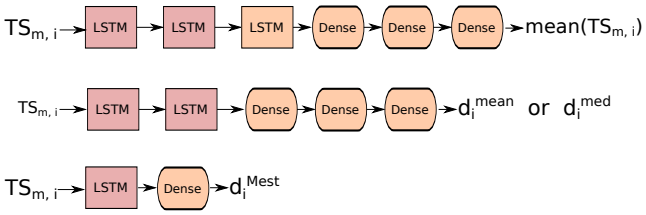

Fig. 2. Mean prediction model

directly for a given segment. We use these three properties in our anomaly test. After finding statistical properties, we determine deviations for every segment as described in the following section.

C. Deviation calculation

Once statistical characteristics of every segment are extracted, we determine the maximum deviation of every segment from the particular statistical properties using LSTM neural network. Thus, we use three other LSTM networks to estimate the deviation from the mean, median, and M-estimator. The deviations are predicted rather than directly computing it because the segment TS_i may contain anomalies. Thus, directly computing deviation may not provide actual deviation values. Moreover, while predicting deviation, we use $TS_{m,i}$ rather than TS_i as the input. Thus our deviation calculation process is not affected by the presence of anomalies.

1) Development of training data for deviation calculation: To better understand the concept of deviation, we describe below, given a segment TS_i how actual deviations are calculated, and what input is fed to the LSTM for training. As we all know, during the training phase, the known inputs and outputs are used. The network is trained using known input and output values. The predicted network output is compared with the known output and based on the difference between the two, the weights of the network are updated.

The known input values used for training the LSTM neural network are $TS_{m,i}$. For developing an LSTM predictor for estimating deviations from the mean, median, and M-estimator, known output values are calculated as depicted in equation 2.

$$d_i^{mean} = max(|x_i - \overline{TS_i}|,\ \forall i \in (1, 2, ...k-1) \quad (2a)$$
$$d_i^{med} = max(|x_i - M(TS_i)|,\ \forall i \in (1, 2, ...k-1) \quad (2b)$$
$$d_i^{Mest} = max(|x_i - S(TS_i)|,\ \forall i \in (1, 2, ...k-1) \quad (2c)$$

Here, d_i^{mean}, d_i^{med}, and d_i^{Mest} are the deviations from mean, median, and M-estimator respectively. We consider M and S as functions to find median and M-estimator values. The LSTM neural network predicts these deviations using clean $TS_{m,i}$ data.

2) LSTM architecture: As described above, we have used 4 LSTM models in the given anomaly detector. One of them for predicting mean as described in earlier in current section IV-B. The other three models are for predicting deviation from the mean, median, and M-estimator. We have optimized the 4 LSTM Neural Network model for the number of LSTM layers and Dense Layers. The models are depicted in Figure 2. The topmost model in Figure 2 describes the LSTM model to predict mean values. Middle model in Figure 2 describes the LSTM models that predict deviation from the mean and deviation from the median. The model at the bottom of Figure 2 describes the LSTM model that predicts deviation from the M-estimator.

D. Anomaly test

We combine the obtained statistical properties and predicted deviation values to test if a given point in a segment is an anomaly or not. We perform three different tests for a given point as depicted in equation 3 subject to the constraints defined for corresponding equations in 4.

$$(x_i < \overline{TS_i} - \alpha.d_i^{mean}) + (x_i > \overline{TS_i}) + \alpha.d_i^{mean}) = 1 \quad (3a)$$
$$(x_i < M(TS_i) - \beta.d_i^{med}) + (x_i > M(TS_i) + \beta.d_i^{med}) = 1 \quad (3b)$$
$$(x_i < S(TS_i) - \gamma.d_i^{Mest}) + (x_i > S(TS_i) + \gamma.d_i^{Mest}) = 1 \quad (3c)$$

s.t.

$$(x_i < \overline{TS_i} - \alpha.d_i^{mean}).(x_i > \overline{TS_i} + \alpha.d_i^{mean}) = 0 \quad (4a)$$
$$(x_i < M(TS_i) - \beta.d_i^{med}).(x_i > M(TS_i) + \beta.d_i^{med}) = 0 \quad (4b)$$
$$(x_i < S(TS_i) - \gamma.d_i^{Mest}).(x_i > S(TS_i) + \gamma.d_i^{Mest}) = 0 \quad (4c)$$

The left-hand side of equations 3 is computed to test if its value is 1 or zero. If the value equates to be 1 for a particular statistics, then the point x_i is classified as an anomaly using that statistic. On the other hand, if its value is 0, then it is not an anomaly. We test for anomalies for all three statistics and then fed the result of different statistic to the counter.

E. Counter

Counter is a simple voter that decides if a point is an anomaly or not. It takes the input from anomaly tests and based on majority it decides whether a point is an anomalous or not. Thus, if any two of the statistics says that the point x_i is an anomaly, then it is classified as an anomaly.

V. Results

The proposed method is implemented in a Python-based framework. The LSTM network is implemented using Keras library. The traffic data is obtained from the Performance Measurement System [17]. We simulated anomalies or outliers by varying the measured sensor data. The simulated anomalies can be either positive or negative. Positive anomalies are those that have values more than the expected value. The negative anomalies have values less than the expected minimum value. The anomalies are added by either adding or subtracting the actual values by $stdDev \times m$, where $stdDev$ is the average standard deviation of every segment in training data, and m

m	P	R	F
1	0.80121	0.39621	0.53022
2	0.88715	0.60413	0.71878
3	0.92648	0.72106	0.81096
4	0.94730	0.78044	0.85581
5	0.96266	0.81470	0.88252
6	0.97003	0.83433	0.89708
7	0.97745	0.84348	0.90554
8	0.98501	0.85230	0.91386
9	0.98586	0.85828	0.91766
10	0.99045	0.86327	0.92250

TABLE I
ANALYSIS FOR POSITIVE ANOMALIES

is a natural number denoting the magnitude of the anomaly strength.

1) Performance metrics: We analyzed Precision, Recall, and F-measure to assess the effectiveness of our anomaly detection model on traffic series data with anomalies inserted in each segment. The three quantities are computed as given in equation 5.

$$P = \frac{tp}{tp + fp} \tag{5a}$$

$$R = \frac{tp}{tp + fn} \tag{5b}$$

$$F = 2 * \frac{P * R}{P + R} \tag{5c}$$

Here, tp is the number of true positives, fp is the number of false positives, and fn is the number of false negatives. The F-measure is calculated by combining the precision and recall and weighting them equally.

2) Performance evaluation: The performance is evaluated by first varying the strength of the anomalies. We analyze the performance both for positive and negative anomalies. We also analyze the performance when the percentage of anomalies is varied. Since, parameters α, β, and γ have been used in proposed anomaly detector, we analyze the performance as these parameters are varied.

A. Analysis of anomaly magnitude

The Tables I-III below illustrate the changes in each of the Precision, Recall, and F-measure metrics when the anomalies of each segment of size $k = 12$ are modified by a value of the given magnitude times the standard deviation of the time segment.

1) Positive anomalies: The performance metrics as the strength of positive anomalies is varied is shown in Table I. We add anomalies in the upper 25% values in each time segment. We measured the Precision, Recall, and F-measure values when the anomaly magnitude was varied from $1-10$ times the average standard deviation $stdDev$ of the segments in training data. The table shows that the Precision, Recall, and F-measure tended to increase as magnitude increases. This is to be as expected as a group of extremely large values will continue to appear quite anomalous in the context of the segment.

m	P	R	F
1	0.65800	0.29890	0.41107
2	0.80999	0.51763	0.63162
3	0.87467	0.65818	0.75114
4	0.90642	0.72821	0.80760
5	0.92798	0.78443	0.85019
6	0.94023	0.81637	0.87393
7	0.95399	0.82418	0.88434
8	0.96076	0.81054	0.87928
9	0.96816	0.79923	0.87562
10	0.97517	0.80356	0.88109

TABLE II
ANALYSIS FOR NEGATIVE ANOMALIES

m	P	R	F
1	0.78044	0.74676	0.76323
2	0.88451	0.80065	0.84049
3	0.93852	0.81512	0.87248
4	0.97286	0.82285	0.89159
5	0.98755	0.83134	0.90274
6	0.99351	0.84007	0.91037
7	0.99676	0.84356	0.91378
8	0.99882	0.84132	0.91333
9	0.99970	0.83683	0.91104
10	1.0	0.83383	0.90938

TABLE III
ANALYSIS FOR POSITIVE AND NEGATIVE ANOMALIES

2) Negative anomalies: We also altered the given time segment by subtracting standard deviations to the lower 25% of values in each time segment. The Precision, Recall, and F-measure values are evaluated as the magnitude of anomalies vary from $1-10$ times of $stdDev$. We observed that Precision, Recall, and F-measure increase as the magnitude increases. The results are recorded in Table II.

3) Positive and negative anomalies: We evaluate the results when both one positive and one negative anomaly are present in Table III. The performance metrics are obtained when the anomaly magnitude is varied from $1 - 10$. We observe that Precision tend to increase as magnitude increases. We also observe that Recall also tend to increase, but it appeared to level out at around $0.83-0.84$. Both results are to be expected as the anomaly test becomes less sensitive to anomalies as magnitude increases, and we see more true positives and fewer false positives. Interestingly, Recall appears to decrease at higher values of magnitude. This may be due to the variability in anomalies being on the threshold of the anomaly tests.

B. Analysis of anomaly percentage

The Table IV illustrates the changes in each of the Precision, Recall, and F-measure metrics when the percentage of anomalies are modified in each time segment. We measure the Precision, Recall, and F-measure values when the anomaly percentage is varied up to 50%. We observed that Precision tended to increase except for when there are 8.3% anomalous values in the segment. We also observed the highest Recall value when there were 16.667% anomalies in the segment. Otherwise, recall tended to be around 0.60, and decreased as the number of anomalies in the segment increased. This

% of anomalies	P	R	F
8.3333	0.76974	0.63224	0.69425
16.667	0.93852	0.81512	0.87248
25	0.95989	0.61294	0.74815
33.333	0.98490	0.60230	0.74748
41.667	0.99270	0.58393	0.73533
50	0.99813	0.57643	0.73081

TABLE IV
ANALYSIS AS PERCENTAGE OF ANOMALIES IS VARIED

suggests a limitation to our algorithm's effectiveness as the statistical methods become less sensitive to more prevalent anomalies.

C. Analysis on Parameter Variation

In this section, we analyze the performance of individual tests as their corresponding parameter α, β, or γ is varied. In our analysis, we measured the variation in Precision, Recall, and F-measure by keeping two tunable parameters constant while varying the third. We varied the third tunable parameter in steps of size 0.25 and starting from 0.5 up to 4.25 while keeping the values of the other two tunable parameters fixed at 2.0. Figure 3 shows the changes in the statistical metrics when the parameter α, corresponding to the anomaly test using the mean, is changed. As shown in the figure, even when the α parameter is low, such that the anomaly test threshold is highly sensitive, we can measure a very high rate of effectiveness with our algorithm, with Precision over 90% and Recall over 80%. Note that as α increases, there is an inverse correlation between Precision and Recall. This is because as the threshold of anomaly classification using the mean becomes higher. Innocuous values are less likely to be classified, leading to fewer false positives, but anomaly values are less likely to lie outside the anomalies test threshold, leading to more false negatives. The best balance between Precision and Recall appears to be when $\alpha = 0.5$, as the F-measure value is the highest at that point.

Figure 4 shows the changes in statistical metrics when the parameter β, corresponding to the anomaly test using the median, is changed. As shown in the figure, we observe an inverse correlation between Precision and Recall, with Recall being higher at low values of β and Precision being higher at high values of β. These changes are much more precipitous than that of the α parameter. Interestingly, Precision appears to decline slightly at high values of β. The best balance between Precision and Recall appears to be at $\beta = 1.5$, where the F-measure is highest.

Figure 5 shows the changes in statistical metrics when the parameter γ, corresponding to the anomaly test using the M-estimator, is changed. We again observe the inverse correlation between Precision and Recall. However, the rate of increase of Precision is much higher than the rate of decrease of Recall than observed in the α and β metrics. As a result, we observe the highest F-measure value at $\gamma = 2.25$, higher than that of the previous two parameters.

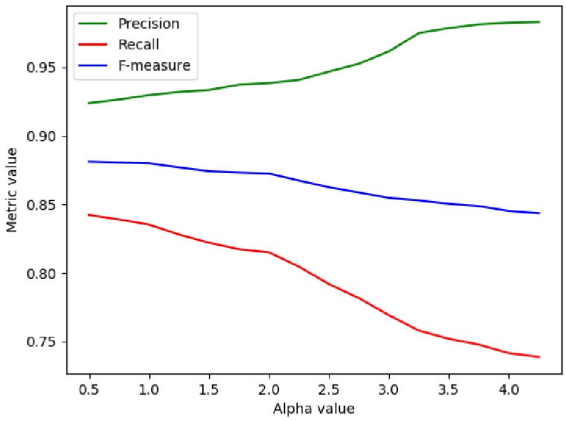

Fig. 3. Performance of mean test as α is varied

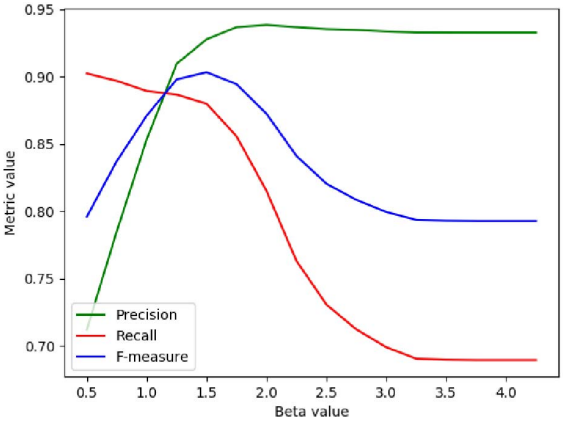

Fig. 4. Performance of median test as β is varied

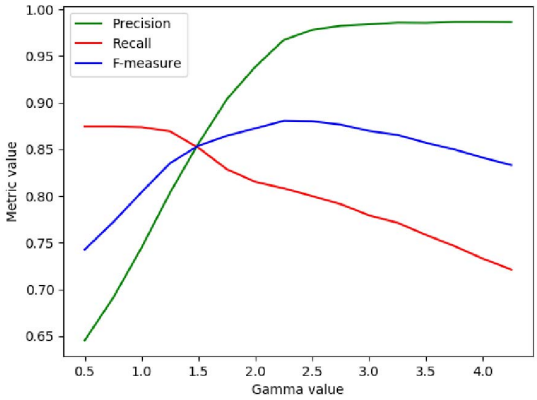

Fig. 5. Performance of M-estimator test as γ is varied

VI. Conclusions and Future work

This paper highlights the challenges behind anomaly detection as it relates to large quantities of time-series data. We presented an automated anomaly detection method that relies on supervised learning and statistical methods to determine anomalies within the time-series. Our proposed anomaly detection method relies on segmentation to determine anomalies within their local contexts. We measured the efficacy of our method from different perspectives, namely variation in tunable parameters, anomaly magnitude, and the number of anomalies, and quantified it in terms of Precision, Recall, and F-measure in each case. Our method does best with smaller tunable parameter values with low yet not insignificant amounts of anomalies. In the future, there are certain following improvisations that could increase the efficacy of the anomaly detector.

1) Scalability: In connected communities, the number of sensors and devices are enormous. Thus anomaly detector not only has to be accurate but also scalable. In this regard, specific clustering methods or transfer learning-based approach could be adopted where a model developed for one sensor can be applied to other correlated sensors. Furthermore, for imparting scalability a hierarchical model can be developed. Thus for the sensors present in a small area, anomaly detection can be performed in edge devices [19]. For the spatially scattered sensors, the processing can be done in cloud.

2) Statistical property matching: Different time-series may have different robust statistical properties. For example, for a specific time-series data, the mean may be a robust statistics, while for other, Median or M-estimator may be a robust value. Thus, before feeding a time-series for statistical analysis, a matching layer can be added that determines the robust statistics for a given sensor. Then only those robust statistics should be used for anomaly detection.

3) Open source: Open-source software, packages, and operating system are common in connected communities as they enable easy to reconfigure, add, or remove devices and applications. Without the open source softwares the devices may go defunct due to unavailability of supported formats [20], [21]. However, due to the open-source software, a malicious adversary may have perfect knowledge of the anomaly detector model. This may help him to devise attack models that can bypass the security mechanism. The analysis and defense mechanism when anomalies are tailored to bypass anomaly detector is an interesting topic to be explored.

Acknowledgement

This research is supported by NSF Award #1723814.

References

[1] R. Shukla, S. Sengupta, and M. Chatterjee, "Software-defined network and cloud-edge collaboration for smart and connected vehicles," in *Proceedings of the Workshop Program of the International Conference on Distributed Computing and Networking*, January 2018, pp. 6:1–6:6.

[2] M. Wu, T. Lu, F. Ling, J. Sun, and H. Du, "Research on the architecture of internet of things," in *Proceedings of IEEE International Conference on Advanced Computer Theory and Engineering (ICACTE)*, vol. 5, August 2010, pp. V5–484–V5–487.

[3] R. M. Shukla, P. Kansakar, and A. Munir, "A neural network-based appliance scheduling methodology for smart homes and buildings with multiple power sources," in *Proceedings of the IEEE International Symposium on Nanoelectronic and Information Systems (iNIS)*, December 2016, pp. 166–171.

[4] R. Shukla, S. Sengupta, and A. Patra, "Smart plug-in electric vehicle charging to reduce electric load variation at a parking place," in *Proceedings of the Annual Computing and Communication Workshop and Conference (CCWC)*, January 2018, pp. 632–638.

[5] J. Hochenbaum, O. S. Vallis, and A. Kejariwal, "Automatic anomaly detection in the cloud via statistical learning," *CoRR*, vol. abs/1704.07706, 2017. [Online]. Available: http://arxiv.org/abs/1704.07706

[6] A. Vempaty, L. Tong, and P. K. Varshney, "Distributed inference with byzantine data: State-of-the-art review on data falsification attacks," *IEEE Signal Processing Magazine*, vol. 30, no. 5, pp. 65–75, September 2013.

[7] R. M. Shukla and S. Sengupta, "Analysis and detection of outliers due to data falsification attacks in vehicular traffic prediction application," in *Proceedings of the IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, November 2018, pp. 688–694.

[8] P. J. Rousseeuw and M. Hubert, "Anomaly detection by robust statistics," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 2, p. e1236, 2018.

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

[10] J. Zhou and A. K. Tung, "Smiler: A semi-lazy time series prediction system for sensors," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, May 2015, pp. 1871–1886.

[11] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 187 – 197, 2015.

[12] T. Salman, D. Bhamare, A. Erbad, R. Jain, and M. Samaka, "Machine learning for anomaly detection and categorization in multi-cloud environments," in *IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)*, June 2017.

[13] Rad - outlier detection on big data. [Online]. Available: https://medium.com/netflix-techblog/rad-outlier-detection-on-big-data-d6b0494371cc

[14] W. C. Lin, S. W. Ke, and C. F. Tsai, "Cann: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowledge-based systems*, vol. 78, pp. 13–21, 2015.

[15] 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, Jun 2016.

[16] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19 – 31, 2016.

[17] The freeway performance measurement system. [Online]. Available: https://people.eecs.berkeley.edu/~varaiya/papers_ps.dir/PeMSTutorial.pdf

[18] R. Shukla and S. Sengupta, "A novel software-defined network based approach for charging station allocation to plugged-in electric vehicles," in *Proceedings of IEEE International Symposium on Network Computing and Applications (NCA)*, Cambridge, MA, USA, 2017, pp. 437–441.

[19] R. M. Shukla and A. Munir, "A computation offloading scheme leveraging parameter tuning for real-time iot devices," in *Proceedings of the IEEE International Symposium on Nanoelectronic and Information Systems (iNIS)*, Dec 2016, pp. 208–209.

[20] Open source iot is growing in importance. [Online]. Available: https://www.iotworldtoday.com/2018/05/24/open-source-iot-growing-importance/

[21] The role of open source in iot. [Online]. Available: https://opensourceforu.com/2017/07/open-source-role-in-iot/