공학석사학위논문

IoT 환경에서 센서 데이터의 이상 감지 기법

Anomaly Detection in Sensor Data in an IoT Environment

2023년 2월

서울대학교 대학원
컴퓨터공학부
박 민 지

IoT 환경에서 센서 데이터의 이상 감지 기법

Anomaly Detection in Sensor Data in an IoT Environment

지도교수 하 순 회

이 논문을 공학석사 학위논문으로 제출함

2022년 11월

서울대학교 대학원

컴퓨터공학부

박 민 지

박민지의 공학석사 학위논문을 인준함

2023년 1월

위 원 장 이 창 건 (인)

부위원장 하 순 회 (인)

위 원 김 태 현 (인)

Abstract

Anomaly Detection in Sensor Data in an IoT Environment

Minji Park
Department of Computer Science and Engineering
College of Engineering
The Graduate School
Seoul National University

For safe operation of an IoT system, it is crucial to detect anomaly in sensor data, which may be caused by various reasons such as device failure, noise, and security attack. In this work, we present an anomaly detection technique reflecting the distinct characteristics of abnormal behaviors in the IoT environment. It consists of two steps. First, we detect the anomaly in each individual sensor data adopting the ARIMA model. To reduce false alarms, we trace the min-max range of normal data in addition. The second step is to detect the anomaly in the combination of sensor data by using a supervised learning technique. Since abnormal combination of sensor data is rarely collected during IoT operation, we devise a novel method to generate artificial outliers specialized in IoT. With the sensor data collected from our laboratory, the proposed technique is evaluated and comparison with the related work shows its viability.

Keywords : Internet of things, anomaly detection, outlier generation
Student Number : 2021-25477

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

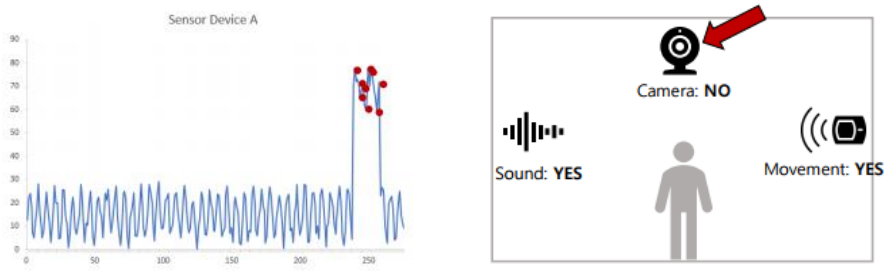
An IoT system determines the environmental situation based on the data collected from the sensors and triggers the user-defined actions of smart devices without requiring the direct involvement of human agents. There is a potential risk that wrong actions can be performed if wrong data is entered to the system. Therefore, for the safe operation of an IoT system, it is crucial to detect anomalies in sensor data, which may be caused by various reasons such as device failure, noise, and security attacks. Anomaly detection is to identify rare events caused by severe deviation from the system's normal behavior. How to treat anomaly data depends on the cause of the anomaly. In case it is caused by a truly abnormal situation, proper actions should be taken to handle the situation. For example, when a fire is detected, immediate action should be taken autonomously by the IoT system. In case of device failure, the user is notified to replace the device, and the sensor value needs to be ignored.

Research on anomaly detection has been conducted for a long time in various fields, including surveillance, manufacturing, financial analysis, and statistics. While numerous methods to detect anomalies have been proposed, they are usually specific to the application domain and depend on the characteristics of the sensor data. The characteristics of IoT sensor data make anomaly detection in IoT more challenging than in other fields [1]. First, it involves a wide variety of sensor devices in terms of types and ranges of sensor

data, and the number of devices is growing. Some sensor devices, such as temperature or humidity sensors, produce continuous values, while devices that generate discrete values, such as motion detection sensors, exist. For the former type of sensors, abrupt value in a short period may be considered anomalous. Anomaly detection of discrete sensor value depends on the system status, on the other hand. Second, low-end sensor devices may produce noisy values that can be regarded as anomalies. To reduce false alarms, it is necessary to distinguish noise from an anomaly. Third, sensor devices may be vulnerable to security attacks. An attacker may trick the system to perform harmful actions by modifying the sensor value at will. We assume that only a few sensor devices can be hijacked by an attacker simultaneously.

In this paper, we propose an anomaly detection technique, considering the aforementioned characteristics of IoT data. The proposed anomaly detection technique consists of two steps. The first step is to detect anomaly in individual sensor data. This process detects anomalies by time-series data analysis, adopting the ARIMA (Auto-Regressive Integrated Moving Average) model. It is a well-known time-series analysis model to forecast the future value based on the history of sensor values even though the sensor data is non-stationary in the sense of mean. We augment the ARIMA model to trace the min-max range of normal data to reduce false alarms. The second step is to detect anomalies in the combination of sensor data. Anomaly in the combination of sensor data here refers to a case where there seems no problem from the perspective of individual sensor data, but the combination indicates that something wrong happens. Suppose that there are a motion sensor, a sound sensor, and an infrared camera in a room. While the infrared camera does not detect a person in the room, both the motion sensor and the sound sensor send data indicating the presence of a person. Then, with a high probability, we can conclude that something goes wrong with the camera: the camera may be controlled by an attacker.

This step, anomaly detection based on a combination of sensor data, is similar to previous methods that have been proposed for contextual anomaly detection or collective



(a) Anomaly in individual sensor data (b) Anomaly in combination of sensor data

Figure 1.1: Example of different types of anomalies

anomaly detection. For instance, Hayes and Capretz [2] proposed a method for detecting contextual anomalies. Similar to the proposed technique, their technique consists of two components: the content anomaly detector and the contextual anomaly detector. The content anomaly detector monitors the individual sensor data with a univariate Guassian predictor for fast detection. Contextual anomaly detection is used to reduce the false alarm when normal data are considered abnormal by the content anomaly detector. The goal of their contextual anomaly detection is different from our second step, even though multiple sensor values are used to define the context of the system. On the other hand, collective anomaly detection methods share the same objective as our second step to detect abnormal situations by considering the collection of sensor data. Deep learning-based techniques have been extensively researched recently for collective anomaly detection [3].

Since it is difficult to label each sensor data accurately and abnormal data is hard to collect in reality, many approaches for collective anomaly detection are based on unsupervised learning. Since our experiments show that the state-of-the-art (SOTA) unsupervised learning techniques fail to achieve a sufficient degree of accuracy for practical use, we propose to use a supervised learning method. To overcome the difficulty of obtaining abnormal data for training, we devise a novel method to generate artificial outliers specialized for IoT. For evaluation of the proposed technique, we collect the normal sensor

data for a month from the in-house IoT system. Experimental results show that the proposed technique improves the classification accuracy significantly higher than the related works, including SOTA techniques. Our key contributions can be summarized as follows:

- We propose an anomaly detection technique composed of two steps: anomaly detection in individual sensor data using the ARIMA model, augmenting with min-max trace analysis, and a collective anomaly detection method based on a supervised learning technique.
- For supervised learning for collective anomaly detection, we propose a novel artificial outlier generation method specialized in the IoT system of interest.
- The proposed technique is evaluated with the real-life sensor data collected from the in-house IoT system. We could achieve above 94% F1 score, which is significantly higher than the other methods could achieve.

The rest of this paper is organized as follows. Chapter 2 reviews the previous approaches for collective anomaly detection and related methods to generate artificial outliers. The proposed anomaly detection technique is explained in Chapter 3, which consists of two steps: anomaly detection in individual sensor data and anomaly detection in the combination of sensor data. Chapter 4 presents the dataset used for experiments and reports the experiment results to verify the viability of the proposed method, comparing it with other approaches. Finally, the paper is concluded in the last section.

Chapter 2

Related Work

Since anomaly detection techniques have been intensively studied for a long time in various fields, the amount of related studies is very large. Since there exist several survey papers available [1, 3], refer to them for a general overview of anomaly detection techniques. In this section, we review the closely related works with the proposed collective anomaly detection method only. In collective anomaly detection, a set of sensor values collected from the associated sensor devices defines a data point.

2.1 Approaches to Anomaly Detection

Since it is unlikely to observe and collect abnormal data points in a real environment, implementing a method to detect collective anomalies should only be done by utilizing collected normal data points. Accordingly, unsupervised learning-based approaches have been widely used for anomaly detection. As a simple unsupervised learning approach, the nearest neighbor approach is to identify anomalies with a distance metric from a data point to its neighbors. K-nearest neighbor (k-NN) and local outlier factor (LOF) are two representative algorithms [4, 5] in this approach. The nearest neighbor approach requires the number of neighbors to be given as a hyperparameter: It is not easy to tune the model. In addition, it is not adequate to deal with a large number of heterogeneous sensor values since the distance metric between two data points is not well defined.

The clustering approach is to group data points into clusters according to a distance or similarity metric among data points. It identifies data points as anomalous when a group contains only a small percentage of the total data [6] or when a data point is far from its nearest group. K-means clustering algorithm and DBSCAN are two well-known algorithms for clustering-based anomaly detection [7]. The clustering approach has a similar drawback as the nearest neighbor approach; it is not easy to tune the hyper-parameters, and it is not adequate to deal with a large number of heterogeneous sensor devices.

Several neural network-based approaches have been extensively researched recently. Reconstruction-based detection with an auto-encoder composed of various types of neural networks is one approach in this category. An auto-encoder is trained only with normal data points to reconstruct an output similar to input data through encoding and decoding steps. Since the network is trained to minimize the error between input and output, the reconstruction error is expected to be small for normal input data. If the reconstruction error is large, the data point is considered anomalous. TranAD [8] is one of the SOTA methods with an auto-encoder structure. It has deep transformer network-based encoders and decoders. Generative adversarial networks (GAN)-based detection has also been employed to solve the anomaly detection problem with a similar approach as auto-encoder, in terms of modeling normal data, but by the adversarial training process. MAD-GAN [9] is based on a GAN constructed with a long short-term memory (LSTM) recurrent neural network (RNN). Because these approaches use only normal data points for training, their performance is sensitive to the characteristics of the abnormal data points used in testing.

Considering sensor data as time-series data, there exist several studies on anomaly detection through time-series analysis. The technique using LSTM [10] is one of those methods for detecting anomalies in time-series data. Approaches to forecast time-series data are also frequently used. It is to predict future data using time series forecasting models, such as ARIMA [11] and Prophet [12], and to classify the data point as anomalous

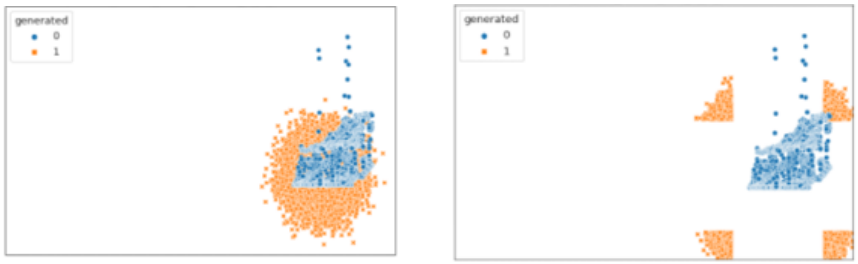
based on the degree of difference between the predicted data and the actual observation.

2.2 Artificial Outlier Generation

Although unsupervised learning-based approaches are predominant in anomaly detection due to the scarcity of actual abnormal data, the different characteristics of data sets impact the performance, and it is often difficult to achieve sufficient accuracy in practice. An alternative to unsupervised learning is to use supervised learning by generating outliers somehow and using them for training. A typical solution to the scarcity problem is data augmentation. Several techniques for time-series data augmentation have been studied [13], considering the unique property of temporal dependency in time-series data. Since only normal data can be collected in an IoT system during normal operation, however, data augmentation could not be a solution to deal with the lack of abnormal data.

In the circumstance of actual abnormal data not being obtainable, generating artificial outliers can be used for supervised learning approaches. Several studies related to artificial outlier generation have been performed [14]. One of the representative methods is density estimation [15]. This method is to generate outliers close to normal instances. It estimates the density of normal data and samples outliers from the distribution. The distribution is assumed to be a multivariate Gaussian distribution since the actual distribution of normal data is unknown. The form of the outliers generated through the density estimation method is displayed in Figure 2.1a. Since it considers values simply sampled from the estimated distribution as outliers, the boundary between normal data and generated outliers is ambiguous. It may make the supervised learning-based model hard to distinguish anomalies.

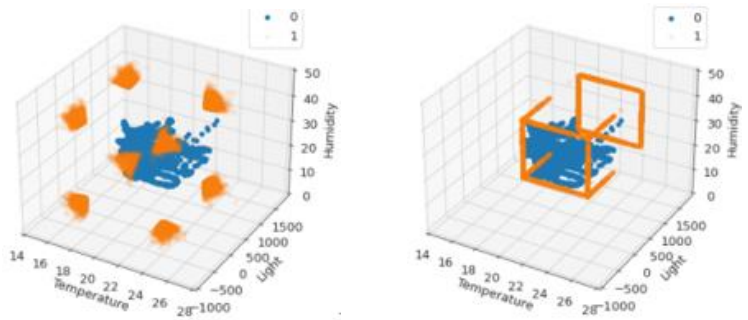
The Gaussian tail method [16] is to generate outliers that are highly different from the normal data. Assuming that the distribution of normal data is a normal distribution, this method samples outliers outside of the range defined by $\mu \pm 3\sigma$ where μ is the mean



(a) Density estimation method (b) Gaussian tail method

(c) Min-max boundary method (d) Sparse regions method

Figure 2.1: 2-D illustration of outlier generation by various methods. Blue points in each figure indicate normal data samples, and orange points indicate outliers generated from each method



(a) Gaussian tail approach (b) Min-max boundary approach

Figure 2.2: 3D illustration of outliers generation from Gaussian tail method and Min-max boundary method

and σ is the standard deviation. The area of generated outliers through the Gaussian tail method is displayed in Figure 2.1b and Figure 2.2a. There is a clear boundary between the generated outliers and the normal data. However, since outliers are generated in a

region too far from the normal data area, it may cause false negatives, which means that the classification model is likely to identify an anomalous data point as normal.

The min-max boundary method [17] is to generate outliers from the min-max boundary of the normal data. The algorithm randomly samples some data points from the normal data set and replaces some sensor values in each sampled data point with their minimum or maximum values. The example shape of outliers generated through the min-max boundary method is as Figure 2.1c and Figure 2.2b. It generates outliers with appropriate boundaries with the normal data but lacks diversity due to simple replacement with minimum or maximum values.

A method to generate outliers in sparse regions [18] is proposed to generate outliers more in a region where the data are rarely seen. The method is to sample data points from the normal data set and switch some sensor values with values that infrequently appear in the normal data set. Figure 2.1d presents the area of outliers generated from this sparse regions method. Outliers generated from this approach also tend to be tough to distinguish from normal data points.

As surveyed in [14], there exist other methods that have different distributions of outliers tailored for the problem addressed in each related work. The four methods reviewed above are selected for comparison with the method proposed in this paper. As explained in Chapter 1, a distinct characteristic of abnormal combinations of IoT sensor data is that only a few sensors may send abnormal data to make the whole data point anomalous. Thus, manipulating a few sensor values in a data point helps to generate outliers suitable for abnormal combinations of sensor data. But most existing algorithms have limitations in generating outliers reflecting this characteristic. Hence we propose a novel method to generate artificial outliers specialized for an IoT system by manipulating a few sensor values only in the sampled normal data points.

Chapter 3

Proposed Anomaly Detection Technique

The proposed technique consists of two steps. The first is to detect anomalies in each individual sensor data, and the second is to detect anomalies in a combination of sensor data. The first step aims to detect the malfunction of a sensor when each individual sensor data shows a trend that is drastically different from the previous trend. In an IoT system, many sensors are fixed at a certain location, periodically sensing and producing time-series data. We can apply a conventional time series analysis method to detect a suspicious condition when a sudden trend change occurs. When an anomaly occurs in a real IoT system, the system alerts a user or the system administrator, and the person in charge would check and take action to restore back to a normal situation. Section 3.1 will discuss the proposed method used in the first step in detail.

After the first step is completed, we perform the second step which is to detect anomalies in the combination of sensor data. Anomaly in the combination of sensor data is a case in which there seems to be no problem from the viewpoint of each individual sensor data, but a problem is suspected when the combination is analyzed. We use a supervised learning method in this stage with generated artificial outliers. Section 3.2 will discuss the proposed method in depth.

The overall flow of the proposed technique is shown in Figure 3.1. We monitor and analyze sensor data in real-time to determine whether there are any anomalies through

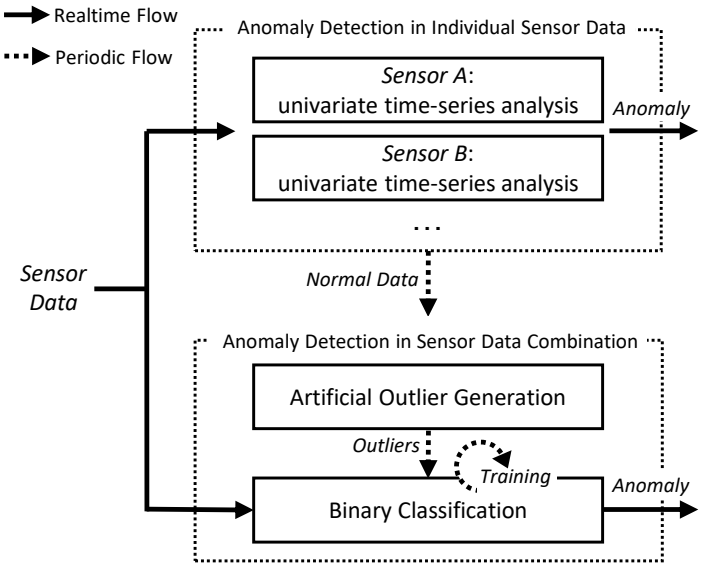


Figure 3.1: Overall flow of anomaly detection

two steps. A univariate time-series analysis is used in the first step and a multivariate classification method in the latter step. For anomaly detection in the combination of sensor data, artificial outliers are periodically generated from the normal data, filtered through the former step, and the classification model is retrained for updates.

3.1 Anomaly Detection in Individual Sensor Data

In order to detect anomalies in individual sensor data, the proposed technique adopts the ARIMA model. It is a well-known time-series forecasting algorithm with widespread usage. Anomaly detection in individual sensor data performs univariate time-series analysis on each sensor data, which shows a trend over time. The ARIMA model can be expressed by the following formula that has three parameters to decide:

$$x'_t = c + \phi_1 x'_{t-1} + \cdots + \phi_p x'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t \tag{3.1}$$

In Equation (3.1), $c + \phi_1 x'_{t-1} + \cdots + \phi_p x'_{t-p} + \varepsilon_t$ represents the autoregressive (AR)

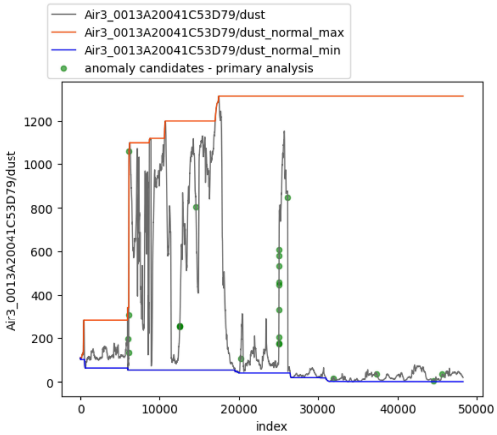
Algorithm 1 Anomaly Detection in Individual Sensor Data

Input : ARIMA parameter p, d, q, window size w, data X
Output : Abnormality A

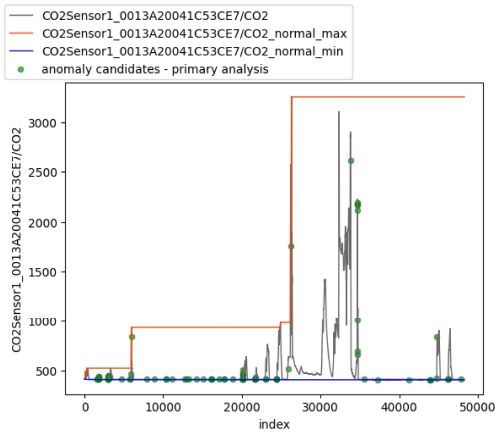
i^{th} attribute data X_i
for $t \in w \ldots$ **do**
  $X_i^{t-1} = [x_i^{t-1}, x_i^{t-2}, \ldots, x_i^{t-w}]$
  Smooth values in X_i^{t-1}
  Normalize X_i^{t-1}
  Fit ARIMA(p, d, q) with X_i^{t-1}
  Predict $\tilde{x}_i^t$
  $e =$absolute errors of X_i^{t-1}
  $threshold = \mu(e) + 3 \cdot \sigma(e)$
  $a_i^t = False$
  if absolute error between $\tilde{x}_i^t, x_i^t > threshold$ **then**
    $a_i^t = reconfirm(x_i^t)$
  end if
  $A_i[t] = a_i^t$
end for
return A_i

model, which predicts the future from the past value. AR parameter p tells how many past time steps would be considered to predict the future with autoregressive coefficients $\phi \in \mathbb{R}$. ε_t is an error term at time t and $\theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$ represents the moving average (MA) model that predicts the future with past error terms. MA parameter q is for past time steps to take into account with moving average coefficients $\theta \in \mathbb{R}$. In the ARIMA model, x', which is differenced value, is used instead of x and the order of differencing, parameter d, indicates that the value at time t would be compared with the value at time $t-d$.

The proposed method is depicted in Algorithm 1. Due to the necessity for online detection in a real environment, we set the window size w and proceed with the moving window analysis. This method performs two preprocessing procedures on the data before ARIMA analysis. One is to reduce the influence of noise, which may be added to the sensor data during a very short period of time. Value smoothing for the data within the window is performed through the moving average, which plays the role of low-pass



(a) Dust sensor3 ($\mu g/m^3$)



(b) CO2 sensor1 (ppm)

Figure 3.2: (a) and (b) shows the suspicious points after ARIMA analysis on values of dust of sensor box3 and CO2 of sensor box1. The lines indicate traced min max values of normal data.

filtering to remove short-lived noise effectively (line 4). Next, we normalize the data after the value smoothing process (line 5). The normalization formula is shown as follows in Equation 3.2:

$$Z = \frac{X - \mu}{\sigma} \tag{3.2}$$

An ARIMA model is fitted with those normalized data, and the data point at time t is predicted (lines 6-7). If the squared error between the predicted data $\tilde{x}_i$ and the actual observation x_i exceeds the threshold, x_i is finally determined as anomalous through an additional reconfirmation procedure (lines 11-12). The threshold for determining abnormality here is set to $\mu + 3 \cdot \sigma$, which is based on the average and standard deviation of the squared errors of the data the model is fitted to (line 9). This procedure is performed within a time window, and the window is moved for the following analysis on new data. The parameters of the ARIMA model from the previous time window are used rather than cold-start with randomly initialized parameters [19] during moving window analysis.

If we apply the ARIMA model naively, false alarms may occur when there is little

Algorithm 2 Individual Sensor Data Anomaly Reconfirmation

Input : ARIMA parameters p, d, q
max = maximum values up to previous time
min = minimum values up to previous time
if x_i^t is in range $[min^t, max^t]$ **then**
  a_i^t = False
else
  Fit ARIMA(p, d, q) with max
  Fit ARIMA(p, d, q) with min
  $eMax$ = absolute errors
  $eMin$ = absolute errors
  $thresholdMax = \mu(eMax) + \sigma(eMax)$
  $thresholdMin = \mu(eMin) + \sigma(eMin)$
  if $x_i^t > max^t + thresholdMax$
    or $x_i^t < min^t - thresholdMin$ **then**
    a_i^t =True
  else
    a_i^t =False
  end if
end if

change in data for a certain period due to the characteristics of the sensor. When the numerical difference of the new observation is relatively larger than that of the data with less variance within the previous time window, although the actual degree of difference is not critically high, false alarms occur. Figure 3.2a and Figure 3.2b display the time-series data collected from two sensors that are used in our experiments: a dust sensor and a CO2 sensor, respectively. Dots on the graph indicate the candidate anomalies that are detected from the ARIMA model. As shown in the figure, the candidates contain many false alarms.

In order to reduce such false alarms, we reconfirm the results in addition to the primary ARIMA-based analysis on sensor values. The $reconfirm$ function in Algorithm 1 performs such procedure. The detail of $reconfirm$ function is described in Algorithm 2. There may be a fluctuation of sensor values in a normal situation due to the non-stationary characteristics of the sensor value, measurement error of the low-end sensor device itself,

or background noise. To reduce the false alarm, we define the allowable range of normal fluctuation. Since the range itself may vary in time, we trace the minimum and maximum values of the normal data to update the range periodically. The reconfirmation procedure traces the minimum and maximum values of normal data. If the data point is within this min-max range, there is a high probability that it is not abnormal (lines 3-4). Thus, data points only exceeding the range are finally judged as abnormal. Since the minimum and maximum values are important criteria for final determination in the reconfirm procedure, we analyze the change of the minimum and maximum values in addition. We perform another ARIMA analysis to find the allowable range of the minimum and maximum values of normal data (lines 6-7). In Figure 3.2a and Figure 3.2b, the red and blue lines represent the trend of the maximum and minimum values of the data up to the previous time of each data point. In order to reduce the false alarms, the data points that exceed this allowable min-max range are finally identified as anomalies among the suspected points (lines 12-13).

In summary, the anomaly detection process in individual sensor data first performs ARIMA analysis on the sensor values and filters out data points suspected as anomalous. After that, if the data points do not exceed the normal min-max range, it is determined as normal. By performing additional ARIMA analysis for the min-max range, the data point which lies outside of the range is identified as anomalous.

3.2 Anomaly Detection in Combination of Sensor Data

To detect anomalies in the combination of sensor data, we use a supervised learning method. Since there is no abnormal data collected from the IoT system, we generate artificial outliers by modifying the sampled normal data points for training. A binary classification model is trained using normal data and the generated outliers. The model is updated by repeating this process periodically. Through the trained model, the newly observed data will be predicted in real-time to check if it is anomalous.

Algorithm 3 Gaussian Random Algorithm

Input : $outNum$, $sensNum$, α, normal data points X
Output : Outliers out

for $i \in 1, \ldots, outNum$ **do**
  Randomly choose $dataPt \in X$
  $out_i = dataPt$
  $sens =$ up to $sensNum$ of sensor values chosen
  for $s \in sens$ **do**
    $min_s =$ minimum of sensor value s in X
    $max_s =$ maximum of sensor value s in X
    $leftBound = min_s - 0.5 \cdot |min_s|$
    $rightBound = max_s + 0.5 \cdot |max_s|$
    $obsValue = dataPt_s$
    $newValue \leftarrow \mathcal{N}(\mu_s, \sigma_s^2)$,
      where $newValue \in [leftBound, rightBound]$,
      where $newValue$ not in range $obsValue \pm \alpha \cdot \sigma_s$
    $out_i[s] = newValue$
  end for
end for
return out

For supervised learning, it is critical for data to be accurately labeled. Since generated outliers affect the performance of the model significantly, we propose a novel method to generate artificial outliers specialized in IoT, which is called *Gaussian random method*. The method reflects the characteristics of the abnormal behaviors in an IoT system; a collective anomaly condition is likely to occur when only a few sensors produce abnormal values while the other sensor values are unaffected. Algorithm 3 displays the pseudo-code of the proposed Gaussian random algorithm.

The Gaussian random algorithm assumes the distribution of each sensor value as a Gaussian distribution. Since We define anomaly in the combination of sensor data as a situation in which each data is within the normal range but the combination is abnormal, the algorithm makes outliers by replacing some sensor values with other values far from the observed value but in the normal range.

The number of sensor values to modify is given as an input parameter *sensNum*

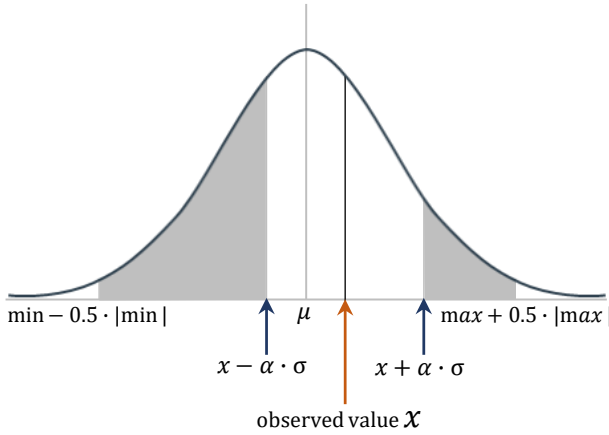


Figure 3.3: The grey area represents the region of sensor values to replace the observed value x to make an outlier by the proposed Gaussian random method

in Algorithm 3 and the number of anomalous data points to generate is given with an input parameter *outNum*. X represents the entire set of the normal data points collected from the IoT system during a given time window. First, a data point to modify is selected randomly from the normal data set (line 2). Then, sensor values to manipulate in the selected data point are also randomly selected (line 4). For each selected sensor, $min - 0.5 \cdot |min|$ and $max + 0.5 \cdot |max|$ are set as the left and right boundaries, respectively (lines 6 to 9). This range is for selecting a new value within the normal range. The algorithm tries to pick a new value from a region within the normal range but far from the observed value. Therefore, the algorithm excludes the area up to $\alpha \cdot \sigma$ away from the observed value *obsValue*. The α parameter allows the algorithm to control how far away from the observed value it samples a new value. The region where the new value can be located is displayed in Figure 3.3. The observed value, x, is replaced with a new value sampled randomly from the grey area. The algorithm generates outliers by repeating this process as many times as the number of outliers to generate (line 1).

After the training data set is formed including the artificially generated outliers, we may use any supervised learning method for binary classification in anomaly detection in the combination of sensor data phase. We use AutoGluon [20], an open-source autoML

framework, to train classification models and compare various algorithms to choose the best one that shows the highest accuracy.

Chapter 4

Experiment

4.1 Dataset

For experiments, a data set is composed of the collected sensor data from our in-house IoT system during the period between June 20th and July 23rd, 2022. As shown in Figure 4.1, the study room (Room A) has a single sensor box and the lounge (Room B) has two sensor boxes, a movement sensor, and a camera. Each sensor box collects environmental information such as temperature, humidity, dust, CO2, sound, and brightness. We may set a different sensing period for each sensor. In the current setting, the period of CO2 sensing is 2 seconds while that of sound and brightness sensing is 5 seconds. The dust level is sensed per 30 seconds. The other sensors send the data only when the value is changed. In the current implementation of the proposed technique, we use a uniform sampling rate of 60 seconds for all sensor data to define a training set for collected anomaly detection. In the case of sound, we choose the maximum value for 60 seconds. For dust, CO2, and brightness, the median value is chosen as the re-sampled value. For the other sensors, the same value is used if no update is made. In summary, a data point consists of twenty sensor values, sampled every minute. The total number of data points is 48,275.

During the data collection period, two sensors showed abnormal behavior as shown in Figure 4.2. Figure 4.2a visualizes the values of the dust sensors in the three sensor

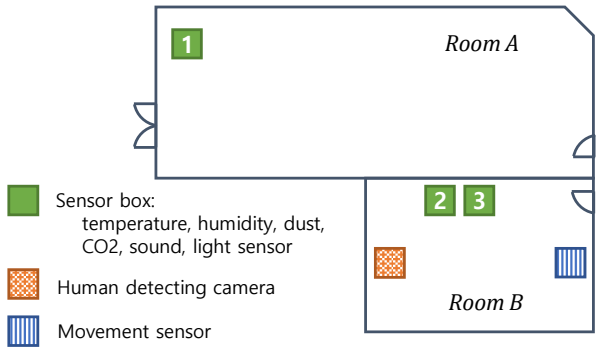


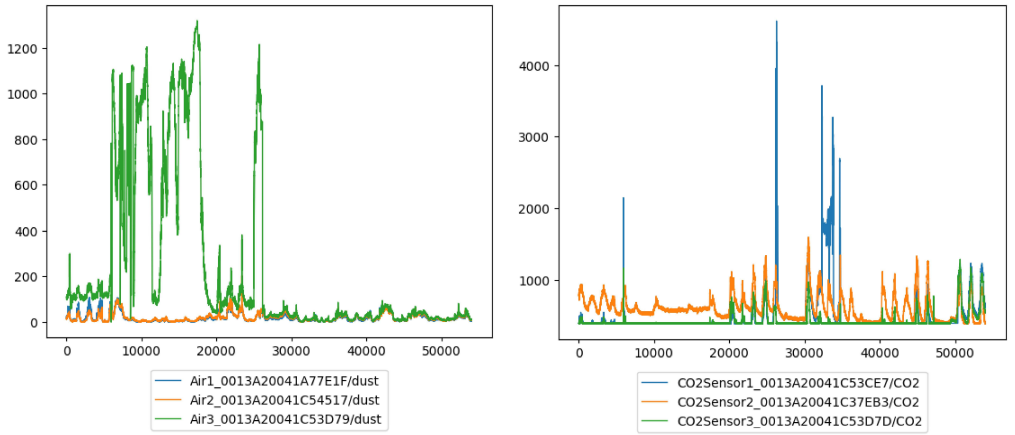
Figure 4.1: Arrangement of sensors in the in-house IoT environment

Table 4.1: Train and Test Sets with Abnormal Scenarios for Anomaly Detection in Sensor Data Combination

	Normal (y=0)	Abnormal (y=1)	Abnormal Scenario
Normal train set	45,395 (100%)	-	-
Test set #1	2,592 (90%)	288 (10%)	camera not detecting person, motion sensor detecting motion
Test set #2	2,592 (90%)	288 (10%)	person detected when the lighting is low, and the sound is low
Test set #3	2,592 (90%)	288 (10%)	temperature and humidity sensors in the same room have different trends
Test set #4	2,592 (90%)	288 (10%)	sound sensor detecting loud sound when the lighting is low

boxes. Looking at the data, the dust sensor in sensor box 3 shows abnormally high values during a certain period, compared with the dust sensor in sensor box 2. Since two sensor boxes are located nearby in the same room, such a difference implies an abnormal situation. Figure 4.2b shows the CO2 sensor values in the three sensor boxes. Occasionally the CO2 level of sensor box 1 shows an abnormal level exceeding 2000 ppm, unlike the other CO2 sensors. We aim to detect the starting point of those abnormal situations for each sensor in the first step of the proposed technique.

All collected data except for those two sensors described above are normal data. After the first step finds the anomalous behavior of two sensors, we form the normal data set that is collected from the other eighteen sensors for collective abnormal detection in the second step. We distinguish the collected data points during the last two days, July 22nd and 23rd, as the test set, and the rest are used as the training data set when training a classification model. An issue is how to define *abnormality* in a combination



(a) Dust sensors ($\mu g/m^3$)

(b) CO2 sensors (ppm)

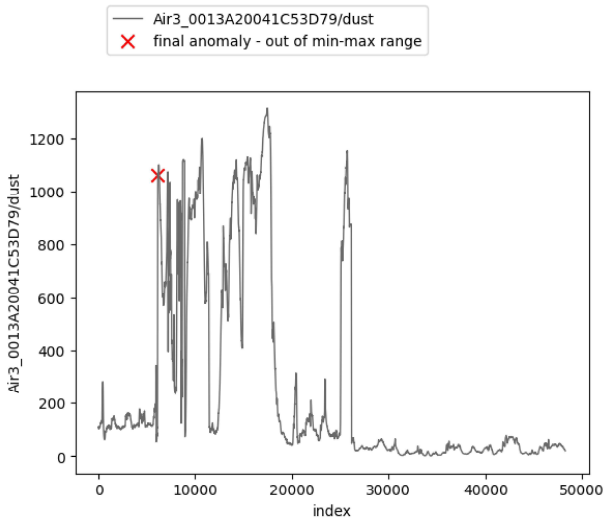
Figure 4.2: (a) displays that the dust sensor of sensor box 3 showed abnormal values (green), and (b) displays that the CO2 sensor of sensor box 1 showed abnormal values for a certain period (blue)

of sensor data. We manually define four abnormal scenarios as presented in Table 4.1 to evaluate collective anomaly detection in the second step. Four test sets are composed by manipulating the data in the normal test set according to four different scenarios. For each scenario, the number of abnormal data points in the test set is set to 10% of that of the normal data points.

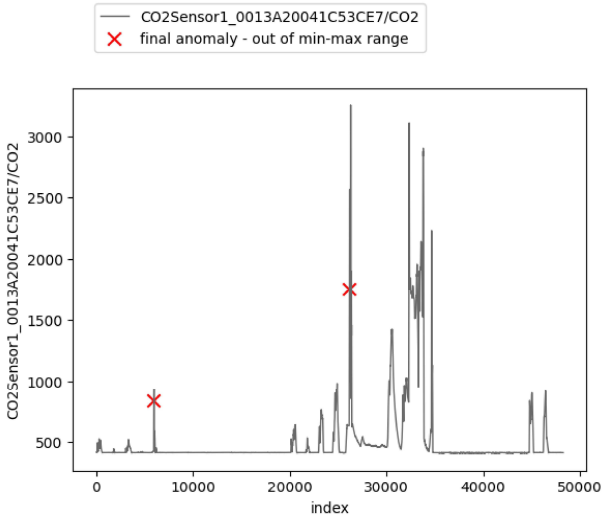
4.2 Individual Sensor Data Analysis

The proposed anomaly detection method for individual sensor data is applied to detect the starting points of abnormal behaviors of the dust sensor in sensor box 3 and the CO2 sensor in sensor box 1, as described in Section 4.1 with Figure 4.2. The AR parameter p, difference d, and MA parameter q of the ARIMA model for analysis on each sensor value are all set to 1. The ARIMA models to analyze the minimum and maximum values of normal data are also set to 1 for each p, d, and q. The window size for moving window analysis is set to six hours.

Figure 4.3a and Figure 4.3b show the results of the final anomaly judgment consid-



(a) Dust sensor3 ($\mu g/m^3$)



(b) CO2 Sensor1 (ppm)

Figure 4.3: Anomaly detection results for two sensor data, Dust sensor3 and CO2 sensor 1 through the proposed ARIMA method augmented by min-max tracing.

ering the min-max range. By augmenting the proposed min-max analysis to the conventional ARIMA model, we could remove all false alarms that are displayed in Figure3.2. Through the experiment, we could validate the viability of the proposed anomaly detection method for individual sensor data, by detecting the starting points of abnormal

behavior of the two sensors that had problems during the data collection period.

4.3 Collective Anomaly Detection

To evaluate the proposed collective anomaly detection method in the second step, four sets of experiments have been conducted. First, we compare the proposed method with unsupervised learning methods. Second, the proposed outlier generation algorithm is compared with other outlier generation algorithms for the supervised learning-based anomaly detection in the second step. Third, we perform a sensitivity analysis of the proposed Gaussian random algorithm to examine how the performance varies as the parameter values change in the proposed Gaussian random method. Lastly, we compare the performance difference between various classification methods.

As a classification model for the first three experiments, we used a gradient-boosting decision tree-based model, LightGBM [21], supported by the AutoGluon framework. This model is chosen since it enables to us to analyze the anomaly detection results and use them in future studies. In the Gaussian random method, we modify at most two sensor values for outlier generation. All the experimental results are statistical results by computing the mean value after ten experiments. F1 score is used as a performance metric that evaluates both recall considering the false negative rate and precision considering the false positive rate.

$$F1\,score = \frac{2 \cdot (precision \cdot recall)}{precision + recall} \tag{4.1}$$

4.3.1 Comparison with Unsupervised Learning Methods

The first experiment is to compare the proposed supervised learning-based method with four unsupervised learning-based methods that include SOTA methods. Selected unsupervised models are LOF[5] as a representative nearest neighbor-based technique,

Table 4.2: Performance Comparison of the Proposed Method with Unsupervised Learning-based Methods

Methods	**test set #1**	**test set #2**	**test set #3**	**test set #4**
MAD GAN [9]	0.0000	0.3733	**0.9574**	0.0000
TranAD [8]	0.2317	0.3120	0.7749	0.8733
Autoencoder	0.7312	0.4642	0.6063	0.8113
LOF [5]	0.9110	0.9240	0.8900	0.0190
Proposed method	**0.9739**	**0.9759**	0.9401	**0.978**

an autoencoder that consists of fully-connected neural layers, MAD-GAN [9] based on a GAN composed of LSTM networks, and TranAD [8] that uses an attention mechanism. These unsupervised learning-based models were trained using the normal training data only. For supervised learning of the proposed method, we generated artificial outliers through the Gaussian random method from the normal train data and include them to train the LightGBM model.

The comparison results are shown in Table 4.2 which reports the F1 score for four test sets shown in Table 4.1. The highlighted F1 scores on the table are the highest for each test set. The proposed method achieves the highest F1 score over 97% on all test sets except for test set #3. Even though it shows the second highest performance on test set #3, the F1 score is more than 94% and the difference from the highest is marginal. It is noteworthy that all unsupervised learning methods result in a large variance of F1 score over the test sets. Even though MAD-GAN gives the best F1 score of about 95.7% on test set #3, it performs worse on all other test sets. LOF shows remarkable performance on three test sets but fails to detect anomaly on test set #4. On the other hand, the proposed method shows little deviation of F1 score on all test sets, while achieving quite high F1 scores. This experiment confirms the superiority of the proposed method to SOTA unsupervised learning methods for collective anomaly detection of IoT sensor data. Note that the performance of supervised learning depends on the training set. The performance gain of the proposed method over unsupervised learning methods is attributed to the proposed method to generate artificial outliers, which will be proven by the next experiment.

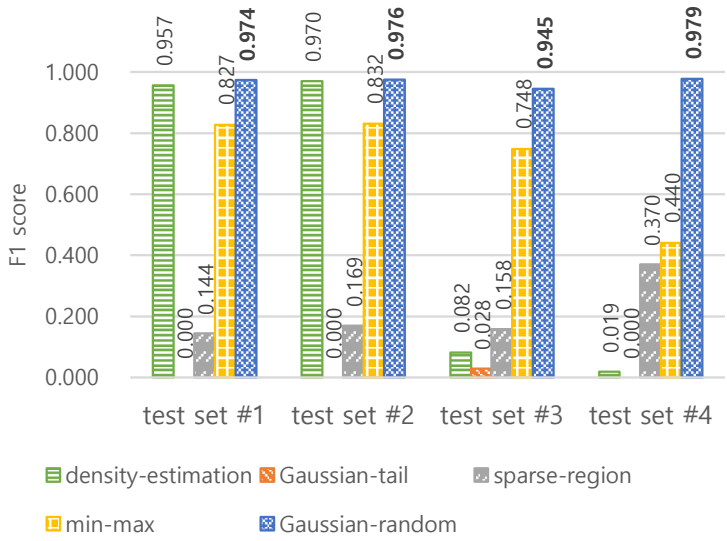


Figure 4.4: Performance comparison of the proposed Gaussian random algorithm with other outlier generation methods in terms of the achieved F1 score by the LightGBM model

4.3.2 Performance of the Proposed Outlier Generation Algorithm

The second experiment is to evaluate the performance of the Gaussian random method that generates artificial outliers. Comparison is made with four outlier generation methods reviewed in Section 2.2, denoted by *density-estimation, Guassian tail, sparse-region*, and *min-max*. Artificial outliers are generated from the normal train data by each method. The LightGBM model is used for supervised learning and it is trained with the train data set that includes both the normal data and the generated outliers.

Figure 4.4 shows the comparison results in terms of the F1 score on four test sets. The figure shows that the LightGBM model achieves higher F1 score when the training set includes the artificial outliers generated by the proposed Gaussian random method than other outlier generation methods on all test sets. In contrast, other outlier generation methods result in poor F1 score or high variance of performance. In case the model is trained with outliers generated by the density estimation method, we observe high variance of F1 score. It induces high F1 score of 95.7% and 97% for test set #1 and test

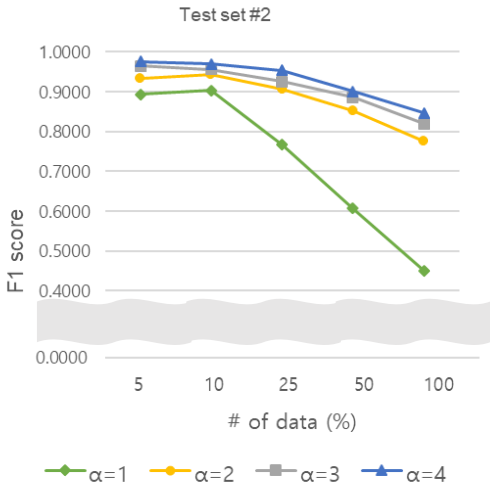
set #2, respectively. But the performance on test set #3 and test set #4 is very poor: F1 score is less than 10%. With the Gaussian tail method that generates abnormal data points far from the normal data points, no meaningful detection result is obtained on all test sets. It is because almost all abnormal data points in the test set are classified as normal. It tells that how to generate outliers affects the F1 score significantly. The sparse region method also performs poorly with F1 score of less than 50% on all test sets. Since the generated outliers are placed close to the normal data points, it is hard to find the classification boundary for anomaly detection.

The min-max boundary method shows a smaller deviation in performance than the other previous methods. For test set #1 and #2, it achieves F1 scores higher than 80%. However, performance on test set #4 is below 50%, which is not adequate for practical use. When the model is trained with the outliers generated by the Gaussian random algorithm, we could achieve high F1 scores from 94.5% on test set #3 to 97.9% on test set #4. It is because the outlier generation mechanism of the Gaussian random method is similar to how the anomalous test set is designed. As explained in Section 4.1 with Table 4.1, an anomalous data point is designed by modifying a few sensor values in a sampled normal data point. It confirms the need for new outlier generation methods for an IoT system, reflecting the characteristics of the IoT data.

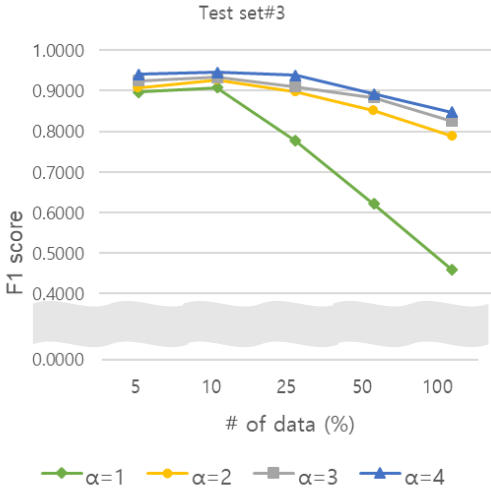
4.3.3 Varying the Parameters of Gaussian Random Algorithm

In the third experiment, we examine how the parameters of the Gaussian random method affect the performance. In the Gaussian random algorithm, the parameter α is a key to control how far away from the original normal data the new value will be sampled, which will replace the original value. The effect of parameter α on the performance is explored by varying its value from 1 to 4. Another parameter is the number of outliers to generate. We investigate how many outliers should be generated and used to train the classification model. The percentage of outlier data points is varied to be 5%, 10%, 25%,

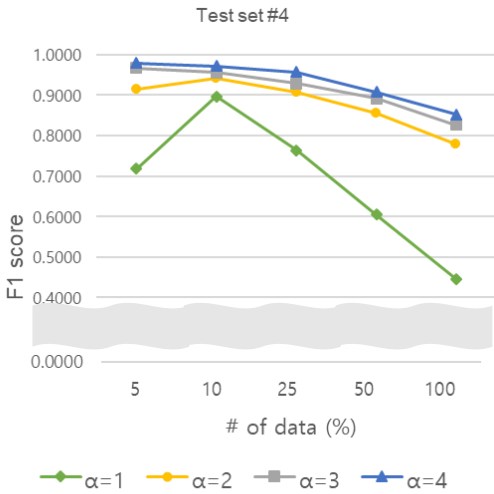
(a) Test set #1



(b) Test set #2



(c) Test set #3



(d) Test set #4

Figure 4.5: Performance variation over the change of two parameters in the Gaussian random method for outlier generation: α and the number of outliers as the percentage of the normal data points.

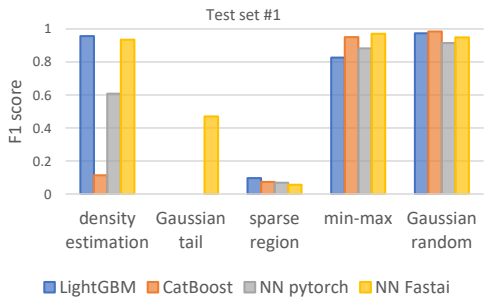
50%, and 100% of the normal data points, respectively. Figure 4.5 reports the experimental results on each test set. The x-axis of each chart represents the percentage of outliers compared to the normal data points. Each graph represents the F1 score of the trained model with a given α value. Four different α values from 1 to 4 are distinguished by colors.

Looking into the figure, we observe that the performance increases monotonically as the α value increases regardless of the number of outliers generated for all test sets. When α is set to 4 rather than 1, which indicates replacing an observed value with a new value from the farther area, the performance gets higher. It means that it is better to place anomaly data points sufficiently far from the normal data points, but definitely closer than the Gaussian tail method. Note that the incremental performance gap decreases as the α value increases.

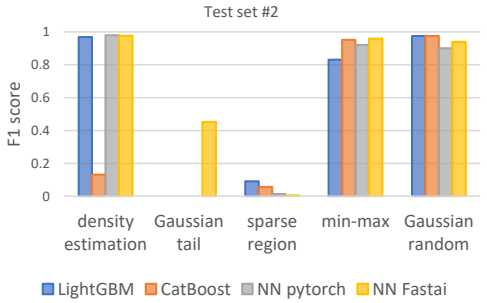
Figure 4.5 also shows how the performance changes according to the amount of generated outliers. We could obtain the highest F1 score when the number of outliers is 5% to 10% of the normal data. The performance decreases as more outliers are generated. We conjecture that too many artificial outliers tend to misguide the classification model to classify normal data points as abnormal. This experiment tells that it is important to control the number of generated outliers in the Gaussian random method.

4.3.4 Performance Difference According to Binary Classification Methods

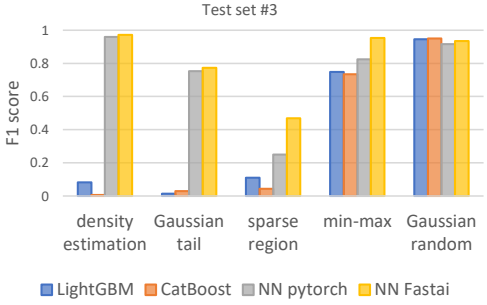
The fourth experiment is to find out whether a binary classification method affects the performance of the proposed method. We evaluate the performance after training different binary classification models after generating outliers by five outlier generation methods used in the second experiment. The following four models are compared as binary classification models: LightGBM, CatBoost model, another decision tree-based model, and two neural net-based models provided by AutoGluon framework. They are



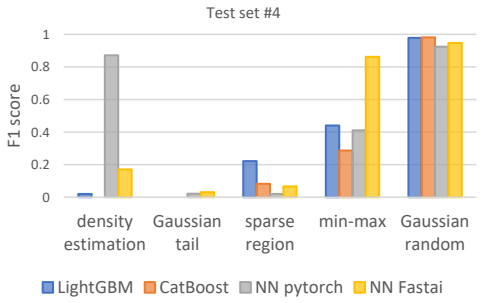
(a) Test set #1



(b) Test set #2



(c) Test set #3



(d) Test set #4

Figure 4.6: Performance difference according to classification methods

selected since they are SOTA models that gave better F1 scores than the other models included in the AutoGluon framework in our preliminary experiments.

The results are displayed in Figure 4.6. Each outlier generation algorithm shows different results depending on the test set and the classification method. The density estimation method gives quite high F1 scores for test sets #1 and #2 when the LightGBM method is applied while very low F1 scores are obtained for test sets #3 and #4. Figure 4.6c and Figure 4.6d show that the performance of the density estimation method is improved on test sets #3 and #4 when a neural net-based classification method is used. The Gauss-tail method fails to classify abnormal data points in all test sets with the LightGBM method as observed above in the second experiment. But if we change the classification method to use neural net-based methods, we could improve the performance significantly except for test set #4. Nonetheless, the obtained F1 score is too low to use in practice.

The sparse region method performs poorly on all test sets regardless of which classification method is used. The min-max boundary method performs well with two decision tree-based models, LightGBM and CatBoost, for test sets #1 and #2 similar to the density estimation method. The performance variation over the test sets and over the classification method is much smaller than the case of the density estimation method. It also performs well with neural net-based methods. The performance of the second neural net-based method on test sets from #1 to #3 could be improved up to F1 score of 95%. It achieves a relatively good F1 score over 85% on test set #4. The min-max method performs best when it is used with the second neural net-based method. Through the above observations, we find that there is a different match between the outlier generation method and the classification method.

Unlike the other outlier generation methods, the proposed Gaussian random method for outlier generation performs consistently well over all test sets and classification methods. We could obtain more than 90% F1 score on all experiments. Even though the difference is not significant, tree-based methods perform slightly better than neural net-based

models. Hence we use the LightGBM method in other experiments.

Chapter 5

Conclusion

In this paper, we propose an anomaly detection technique consisting of two steps, considering the characteristics of IoT data. The first step is to analyze univariate time-series data using the ARIMA model to detect anomalies in each individual sensor data. To reduce false alarms, we augment the ARIMA model by tracing the min-max range of the normal data. The second step is to perform collective anomaly detection by analyzing the combination of sensor data. To this end, we use multivariate classification using an existing supervised learning technique since SOTA unsupervised learning techniques fail to achieve sufficient performance for practical use. To cope with the difficulty of obtaining abnormal data for training, we devise a novel method, called Gaussian random method, to generate artificial outliers, taking into account the characteristics of the IoT data.

The proposed technique is evaluated with real-life sensor data collected from the in-house IoT system. In the first step, we could successfully detect the starting points of abnormal behavior of two sensors, which occurred during the collection period. To evaluate the proposed collective anomaly detection methods in the second step, we designed four test sets of anomaly situations manually. Through extensive experimental results, the following observations could be made. First, the proposed technique performs significantly better than SOTA unsupervised learning techniques on all test sets. Second, the

proposed Gaussian random method for outlier generation performs better than the other outlier generation methods. Third, it is necessary to explore the parameter values of the Gaussian random method to get the highest F1 score. Last, we may use any existing classification method in the second step since the generated outliers reflect the characteristics of the real abnormal behaviors. We plan to release our collected sensor dataset used in our experiments.

Bibliography

[1] Andrew A. Cook, Göksel Mısırlı, and Zhong Fan. Anomaly detection for iot time-series data: A survey. *IEEE Internet of Things Journal*, 7(7):6481–6494, 2019.

[2] Michael A Hayes and Miriam AM Capretz. Contextual anomaly detection in big sensor data. *2014 IEEE International Congress on Big Data*, pages 64–71, 2014.

[3] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[4] Anuroop Gaddam, Tim Wilkin, and Maia Angelova. Anomaly detection models for detecting sensor faults and outliers in the iot - a survey. *2019 13th International Conference on Sensing Technology (ICST)*, pages 1–6, 2019.

[5] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, page 93–104, 2000.

[6] Salima Omar, Asri Ngadi, and Hamid H. Jebur. Machine learning techniques for anomaly detection: an overview. *International Journal of Computer Applications*, 79(2):33–41, 2013.

[7] Dingsheng Deng. Research on anomaly detection method based on dbscan clustering algorithm. *2020 5th International Conference on Information Science, Computer Technology and Transportation (ISCTT)*, 2020.

[8] Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. Tranad: Deep transformer networks for anomaly detection in multivariate time series data. *arXiv preprint arXiv:2201.07284*, 2022.

[9] Dan Li and et al. Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. *International conference on artificial neural networks*, pages 703–716, 2019.

[10] Pankaj Malhotra and et al. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*, 2016.

[11] Viacheslav Kozitsin, Iurii Katser, and Dmitry Lakontsev. Online forecasting and anomaly detection based on the arima model. *Applied Sciences*, 11(7):3194, 2021.

[12] Sean J. Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

[13] Qingsong Wen and et al. Time series data augmentation for deep learning: A survey. *arXiv preprint arXiv:2002.12478*, 2020.

[14] Georg Steinbuss and Klemens Böhm. Generating artificial outliers in the absence of genuine ones—a survey. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(2):1–37, 2021.

[15] Kathryn Hempstalk, Eibe Frank, and Ian H. Witten. One-class classification by combining density and class probability estimation. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 505–519, 2008.

[16] Truong Son Pham, Quang Uy Nguyen, and Xuan Hoai Nguyen. Generating artificial attack data for intrusion detection using machine learning. *Proceedings of the Fifth Symposium on Information and Communication Technology*, page 286–291, 2014.

[17] Chi-Kai Wang, Yung Ting, Yi-Hung Liu, and Gunawan Hariyanto. A novel approach to generate artificial outliers for support vector data description. *2009 IEEE International Symposium on Industrial Electronics*, pages 2202–2207, 2009.

[18] Wei Fan, Matthew Miller, Sal Stolfo, Wenke Lee, and Phil Chan. Using artificial anomalies to detect unknown and known network intrusions. *Knowledge and Information Systems*, 6(5):507–527, 2004.

[19] Viacheslav Kozitsin, Iurii Katser, and Dmitry Lakontsev. Online forecasting and anomaly detection based on the arima model. *Applied Sciences*, 11(7):3194, 2021.

[20] Nick Erickson and et al. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

[21] Guolin Ke and et al. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

요 약

IoT 시스템의 안전한 운영을 위해서는 기기의 고장, 노이즈, 보안 공격 등 다양한 원인에 의해 발생할 수 있는 센서 데이터의 이상 징후를 감지하는 것이 중요하다. 본 연구에서는 IoT 환경이 가진 비정상 상황의 고유한 특성을 반영한 이상 탐지 기법을 제시한다. 제안하는 이상 탐지 기법은 두 단계로 구성된다. 먼저 ARIMA 모델을 적용하여 개별 센서 데이터에서의 이상 징후를 탐지한다. 이 때, 오경보를 줄이기 위해 데이터의 최소-최대 범위를 추가적으로 추적한다. 두 번째 단계는 지도 학습 기법을 사용하여 센서 데이터 조합의 이상을 감지하는 방법이다. 실제 센서 데이터의 비정상적인 조합이 IoT 시스템의 운영 중에 수집되는 경우는 드물기 때문에 IoT에 특화된 인공적인 이상 값을 생성하는 새로운 방법을 고안하였다. 또한, 연구실에서 수집한 센서 데이터를 통해 제안한 기법을 평가하고 관련 연구와 비교하여 가능성을 확인하였다.

주요어 : 사물인터넷, 이상감지, 이상데이터 생성

학번 : 2021-25477