# Data Reduction Approach Based on Fog Computing in IoT Environment

Rawaa Majid Obaise
*Dept. of Computer Science*
*College of Science for Women*
*University of Babylon*
Babylon, Iraq
rawaa.almamuri@gmail.com

Mahdi Abed Salman
*Dept. of Computer Science*
*College of Science for Women*
*University of Babylon*
Babylon, Iraq
mahdi.salman@uobabylon.edu.iq

Hussein A. lafta
*Dept. of Computer Science*
*College of Science for Women*
*University of Babylon*
Babylon, Iraq
wsci.hsein.attia@uobabylon.edu.iq

***Abstract*— This paper investigates a data processing model for a real experimental environment in which data is collected from several IoT devices on an edge server where a clustering-based data reduction model is implemented. Then, only representative data is transmitted to a cloud-hosted service to avoid high bandwidth consumption and the storage space at the cloud. In our model, the subtractive clustering algorithm is employed for the first time for streamed IoT data with high efficiency. Developed services show the real impact of data reduction technique at the fog node on enhancing overall system performance. High accuracy and reduction rate have been obtained through visualizing data before and after reduction.**

***Keywords—Fog computing, IoT, Cloud computing, Subtractive Clustering, Data Reduction***

## I. INTRODUCTION

Lately, the proliferation of IoT and a substantial increase in the internet-connected devices have resulted in a voluminous amount of data often described under the concept of "big data". In this regard, the neediness for an efficient computing paradigm to deal with these data is very essential. Previously, such data is transmitted to the cloud data center for processing, analysis, and storage. As the data velocity and size increase, transferring the big data to the cloud datacenter became ineffective or in some cases impractical due to bandwidth limitations[1, 2]. Also, many IoT applications require real-time data processing that imposes a new challenge on current IoT architectures, especially in meeting the latency, privacy, and bandwidth consumption[3]. So here comes up the need for data reduction and pre-processing directly close to the data source[4]. Primary data processing at fog computing provides timely response to the end devices at local networks and also decreases the amount of data transmitted to the cloud platform to avoid network bandwidth and reduce the cloud computing space in storing redundant or meaningless data[5].

However, several data reduction techniques can be to reduce the stream IoT data transmitted to the cloud. Data reduction aims to obtain a smaller volume of data to represent the actual phenomena that are producing the same analytical results[6]. From a data mining perspective, there are two directions to reduce data size: **Dimensionality reduction** and **Numerosity reduction**. The former drops columns (attributes) from the dataset that have fewer effects on the analyzing result e.g. wavelet transforms, Principal Components Analysis (PCA), feature subset selection, and feature creation, etc. While the latter drops rows (values) from the dataset that are highly redundant in their columns e.g. regression and log-linear models, histograms, etc.

We choose to develop our model in the second direction specifically in the clustering category. According to stream data characteristics, the traditional clustering algorithms became unable to meet the requirements of stream data. Therefore, the neediness to discover or employ clustering algorithm suits stream data characteristics is becoming more and more urgent[7]. One of our objectives is employing subtractive clustering for stream data clustering. Subtractive clustering algorithm [8-12]is adaptive in the way that the cluster center generated, a one-pass algorithm that does not impose the number of initial clusters in advance should be employed.

This paper investigates a data processing model to reduce the size of streamed IoT data transmitted to the cloud using a stream clustering technique. In our model, the subtractive clustering algorithm is employed for streamed IoT data with high proficiency. Developed services show the real impact of data reduction technique at the fog node on enhancing overall system performance. High accuracy and reduction rate have been obtained through visualizing data before and after reduction.

The main contributions provided by the paper can be summarized as follows: 1) We are the first who use the subtractive clustering algorithm for data reduction with IoT data stream. 2) As we show in the comparison with a chosen related works, our model produces a high reduction rate while preserves a high accuracy in the resulting representative data.

The rest of this paper is organized as follows: The related works are in section II. Section III describes the proposed data reduction model. Next, the results are in IV followed by the conclusion in V.

## II. RELATED WORKS

Reducing big IoT data transmitted to the cloud is one of the most important topics, here are some of the literature that employed different reduction techniques with the fog computing concept. The authors in[13], employed a sensors data fusion technique to reduce the enormous amount of data and extract features in IoT devices. More explicitly, in the fog approach, the processing (data fusion and machine learning algorithms) was done entirely in the fog node. While in the hybrid method, the raw data were first fused into the fog node to extracts features, then these features were sent to the cloud

to apply the machine learning algorithms. The authors recommended that both network-level and cloud- level processing work together to create effective IoT data analytics that would address cloud computing limitations.

The authors in[14], proposed a smart fog computing based on a clustering technique to discover trends in morbidity speech data collected from patients with Parkinson's Disease (PD). PD patients use smartwatch while doing home-talking exercises. Speech features have been normalized and processed in fog node using a k-means clustering algorithm. Whenever an abnormal change in the features the results will be uploaded to the cloud.

In[15], proposed a compressive sensing(CS) framework to monitor and classify ECG signals. The system architecture is composed of three parts: data collection using wearable devices, where ECG data were gathered and compressed by performing CS techniques. Data processing utilizing a smart gateway that first reconstructs and then sends the original ECG signals to the cloud datacenter. Cloud computing was used to monitor the patient's medical records for a long time. Also, to assist doctors with the computer-aided diagnosis. However, compression reduces data for communication and storage but not for analyzing purposes.

In [5]proposed a novel temporal data reduction scheme by leveraging the primary data processing at the fog platform to reduce IoT data transmitted to the cloud. The proposed scheme involves two major phases: **1)** IoT data was initially represented as a multivariate normal distribution (MVN) based on historical data at the cloud platform. **2)** Kalman filters (KF) were used at both cloud and fog levels, which concurrently predict and update the mean vector and covariance matrix of the MVN model. Hence, only the measured data out of the predicted range will then be transmitted from the fog node into the cloud. Otherwise, the predicted values will be used at both platforms instead of the actual measurements.

The work in[16], introduced a two-layer compression scheme to reduce the size of IoT data transmitted to the cloud to avoid high bandwidth usage. The scheme proposed was designed to compress numerical IoT sensors data. Firstly, data collected from the sensors is initially compressed in the fog node. Where the collected data was first sorted in ascending order using a sorting algorithm. After that, the mean vector of every pair of data values was computed and rounded. Then, the compressed data was transmitted to the cloud for further compression. At the cloud level, the compression was achieved by counting the frequencies of each value from the data acquired after the initial compression. This manner reduces the data size by keeping only the frequencies of values.

In-networking data reduction technique to reduce the transmitted IoT data to the cloud datacenters has been adopted by authors in[4]. The proposed method includes two layers: filtering and data fusion layer. The main goal of this work is to minimize the data transferred from the edge of the network to the cloud datacenter.

The work in [17], proposed a two-tire data reduction(TTDR) to reduce data transmitted from IoT devices. Data reduction was applied to two-tier: sensor node and gateway. In the former, a simple and suitable data compression method was applied due to the constraints in sensor nodes. The later used data reduction technique (hierarchical clustering) for grouping data received from sensor nodes based on the Minimum Description Length (MDL) principle. The technique employed at the gateway tier depends on the fact that if any pairs of data obtained from the first tier can be compressed by MDL, they will be combined into one cluster. As a consequence, the data sets numbers will be reduced, and the merging of the sets will be terminated when there is no further pair of sets that can be compressed.

## III. PROPOSED DATA REDUCTION MODEL

We developed a model to collect and process streaming data from IoT devices using fog-computing concepts. A testbed has been built is composed of room occupancy monitoring devices, a fog node, and a cloud channel. Each device posts its readings to the fog node in short intervals i.e. 15 seconds (mandatory by free cloud service). When the fog node is unavailable, the devices directly send their readings to the cloud service (cloud acts as a fail-over backup destination). Devices are configured to continually provide a local edge server (fog node) with data about room state. A set of services are deployed on the edge server to process received data before sending abstracted data to the cloud. The fog node behavior (which is considered the heart of the system is described by algorithm 1).

**ALGORITHM 1: FOG-NODE BEHAVIOR**

***Input:*** *fogPeriod, sensorInterval*
***Output:*** *Transferring result of subtractive clustering to cloud channel*
***Initialization:***
    ***Set database connection***
    ***Set cloud connection***
    ***Set startTime*** *// starting point of time for collection*
***Loop***
***dataset=getBlock(startTime,fogPeriod,sensorInterval)***
*// **dataset** is a matrix of n=number of vectors by m=number of sensors.*
*// **fogPeriod** is the length of the period to check the time of the last data item.*
*// **sensorInterval** represents a short interval where all sensor readings are aggregated to one value*
    ***Print(dataset.features())*** *// statistical measures of each column( min,max, mean,...etc.)*
    ***NormalizeData=normalize(dataset)***
    ***Centers=SubtractiveClustering(NormalizedData)***
*// centers are the reduced matrix of the original dataset; each row is a cluster center.*
    ***sendToCloud(Centers)*** *//send centers to cloud*
    ***wait(fogPeriod)*** *// wait for data of next period to be buffered*
    ***set startTime=StartTime+fogPeriod*** *// shift starting point of collection by period length*
***End loop***

The fog node accommodates most processes where it hosts a data collection service through a web-based database. Also, it deploys a developed service to fetch blocks of data from the database and process them by applying the subtractive clustering algorithm periodically. A fog node computes and sends only representative data to the cloud

service instead of raw data. These services are described in the following:

### A. Data collection

Edge server listens to the posts of IoT devices. Using HTTP protocol, data received from these devices are stored in a buffering database through a script embedded in a designated web page. Also, IoT data is displayed on another web page as a series of readings from different devices. The set of tuples that represent old duration may be dropped after they have been processed or kept for process performance analysis.

### B. Data set preparing

The stream data from IoT devices stored in the buffering database. We convert these data from the stream form to a set of vectors that are combined in a block to deal with it easily for processing. Algorithm 2 illustrates the process of creating a vector from streaming data. In detail, data is fetched from the buffering database for a given period. A period is divided into many small intervals. For each interval, the reading for each device is stored in the predefined position of the state vector. If many readings exist, the mean value is computed. The state vector is composed of computing one value for each device. The state vector is a short duration representation (1 minute) for a set of devices. A block represents the room state for a long duration (1 hour).

**ALGORITHM 2: GET-VECTOR**

```
Require: startTime, Interval
// startTime is point of time where the state of a given device is
      read
// Interval represents a slice of time where all device readings
      are aggregated to one value.
Ensure: Vector is complete
Initialization:
sensorVector← {sensor1, sensor2…. sensor n} // devices names
      for refrencing.
rep [] ← {0, 0,…0} // store number of reading for each device
      within the given interval
vector [] ← {0, 0,…0} //actual state of the room for given
      interval
1: Connect with database // establish connection
2: Query database by select * from Logs-Table
where timeStamp > startTime and timeStamp < (startTime +
      Interval)
3: For each data item in the dataset do
4:       For i ←0 to length (sensorVector) do
5:            If sensorVector(i) =item. SensorName then
6:                 Vector[i] ← Vector[i] + item.value
7:                 rep[i] ←rep[i]+1
8:            End if
9:       End for
10:  End for
11: For i ←0 to length (sensorVector)
12:           If rep[i] > 0 then
13:                Vector[i] ← Vector[i] /rep[i]
14:           Else
15:                Return null
16:           End if
17:  End for
18: Return vector
```

Finally, as illustrated in algorithm 3 a set of vectors are generated from n of intervals will form a block. The process of creating a block of vectors from the stream data is modeled by algorithm 3.

**ALGORITHM 3: GET-BLOCK**

```
Require: startTime, endTime, fogPeriod, sensorInterval
Ensure: Block of vectors
1:  While starTime < endTime do
2:    Vector=_GetVector (startTime, fogPeriod, sensorInterval)
3:       if Vectorvalue ≠ null then
4:            add vector to the block
5:            startTime← startTime+ sensorInterval
6:       End if
7: Return block of vectors
8:  End while
```

### C. Data Normalization

Data from IoT devices fall in different ranges so we used Min-Max normalization to scale it into one range [0,1]. Normalization prevents large-range attributes from outstripping attributes within a small range.

### D. Data Reduction

We used a clustering technique to reduce the stream data transmitted from IoT devices to the cloud. Clustering is the most common form of unsupervised learning in data mining[18]. It partitioning data set into several clusters based on similarities (distances) between objects. So, we say that the group of objects is a cluster when the distance between objects in the same cluster is minimum and the distance from other objects in other clusters is maximum.

However, different algorithms can be used to perform clustering process. In general, some of clustering algorithms are based on knowing the number of clusters at the dataset. In this type, the algorithm attempts to partition the data set into a defined number of clusters. Fuzzy C-means and K-means are an example of this type. In other cases, it is not important to know the number of clusters from the beginning, but rather the algorithm begins by finding clusters in sorted order, like the first large cluster, and then proceeds to find the second cluster, and so on. Subtractive and Mountain clustering are of this kind[19].

We used the subtractive clustering algorithm as a stream clustering algorithm. The subtractive clustering algorithm is implemented periodically i.e. every hour on each normalized block. The size of the block is fixed to 60 vectors per hour. Then, only representative data that resulted from each block is sent to the cloud service instead of raw data.

Finally, at cloud level, we used an IoT platform that considers the back-end layer composing of free hosting IoT services, e.g. ThingSpeak, to archive data sent from the fog node. Actually, we have nothing to implement on the cloud except receiving data from either fog-node or directly from IoT devices when the former is out of reach. ThingSpeack provides two services; the first is archiving the representative data without redundancy. Archived data may be used for long-term analysis. The second is real-time visualization that is useful for monitoring from over the Internet.

### E. Performance Evaluation

Our main objective is to reduce the amount of data sent from the IoT devices to the cloud without scarifying the semantic of the data. We need an indicator to show us how much of the original data is drooped at the fog node and how is the data meaning is preserved. The impact of data processing at fog node on the communication between IoT devices and remote site (cloud computing) is evaluated using two measures: **Reduction rate** and **Accuracy**, each one is explained as follows:

*a) Reduction Rate*: The percentage of data reduced by applying clustering techniques (subtractive clustering algorithm) on each block. It is computed by the following formula:

$$\text{Reduction rate}=\left(1-\frac{C}{V}\right)\times 100 \quad (1)$$

Where: **C** is the number of centers resulting from clustering and **V** is the total number of vectors in the input block.

*b) Accuracy:* The accuracy indicates how the center points are useful enough to represent the original data without any loss data variation form. It is computed using statistical measures e.g mean, standard deviation. These measures are applied on raw and representative data to see if, the reduction process corrupts the overall shape of the data.

## IV. EXPERIMENTAL RESULTS

This work experimental environment involves four types of sensors that are deployed in a room. They are Temperature, Humidity, Light, and Sound. Each sensor is integrated with a NodeMCU ESP8266. ESP80266 is a low-cost Wi-Fi microchip with TCP/IP stack and microcontroller. ESP80266 acts as a client sending room status data to a local fog node via HTTP protocol. As a fog node, we used a laptop with the operating system Windows 7, Processor Intel (R) Core (TM) i5-2450M CPU @ 2.50 GHZ(4CPUS), 2.5GHZ, and memory size(RAM): 4GB. A free hosting cloud service (ThingSpeak) was used as a cloud platform.

This section shows the results obtained from implementing the data reduction model that has been presented earlier. Data is collected for duration 2-8 March 2020 for a room in an occupied house in Iraq. We consider one minute for the sensor interval and one hour for fog duration. Fig. 1(a) shows the temperature of the room is varying over days but it keeps a relatively constant range (30-34). The change in the temperature readings is very slight and relatively increasing in spaced intervals. After data is reduced, Fig. 1(b) we can see that the representative data produce relatively similar shape to the original data. Hence, data is still informative and only redundant data was removed.

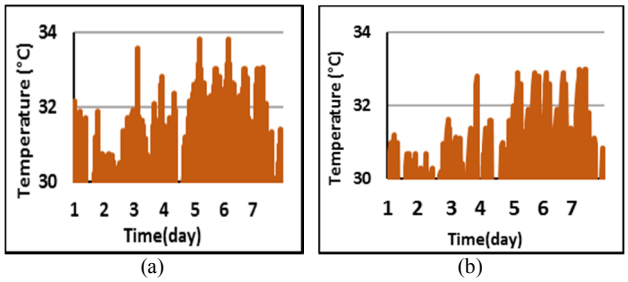



Fig. 1. Temperature data: a) Raw data, b) Representative data

In Fig. 2(a), we see many long-term humidity fluctuations occur over the days. As in temperature readings, the changes in humidity readings are little and become observable in spaced intervals. In general, humidity readings maintain a constant range (25-35). The results in Fig. 2(b) demonstrate that the resulting center points are useful enough to represent the original data without any loss data variation form i.e. data is still insightful and only redundant data was dropped.

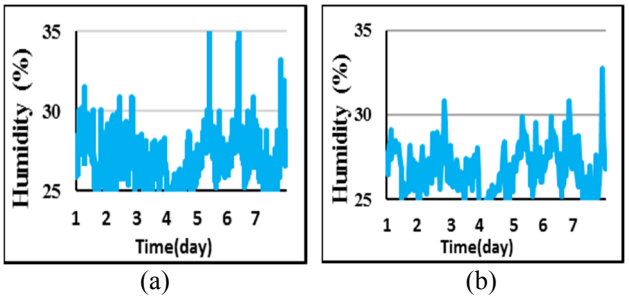



Fig. 2. Humidity data: a) Raw data, b) Representative data

As illustrates in Fig. 3(a) the light of the room continuously changes over the days. In comparison with temperature and humidity sensor, the changes in light sensor readings are occurring within very convergent intervals. The yielded representative in Fig. 3(b), is completely identical to the original data Fig. 3(a) i.e. data is still informative with redundant data removal.

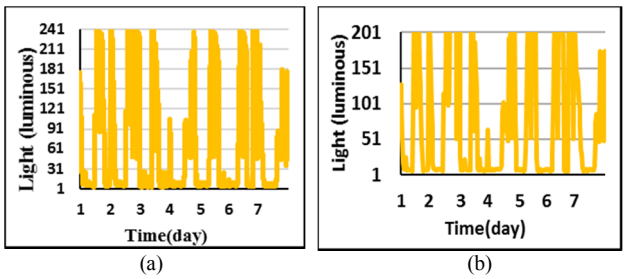



Fig. 3. Light data: a) Raw data, b) Representative data.

The state is slightly different concerning the sound sensor shown in Fig. 4(a). The readings of the sound sensor are always limited within a range between (0-2). Since it is an event-driven sensor that depends on the existence of sound (2) or not (0). As illustrated in Fig. 4(b) the obtained representational data is similar to the raw data shown in Fig. 4(a) and only redundant data was dropped.

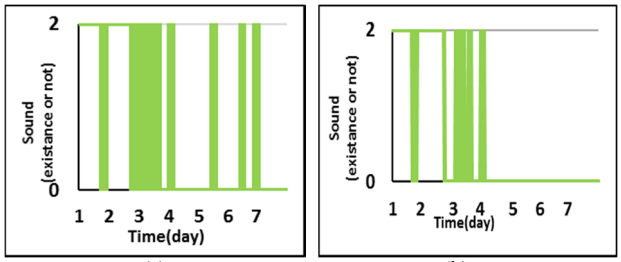



Fig. 4. Sound data: a) Raw data, b) Representative data

As previously mentioned the system performance was evaluated using reduction rate and accuracy measures. As demonstrated in Fig. 5 the reduction rate changes with time. It depends on the number of centers that resulted from each block. Based on "(1)" the average reduction rate obtained up of **97%**. It is noted that the reduction rate in days **(2,4)** is comparatively lower than the rest of the days. This because the reduction rate is reversely commensurate with the variability in data, high variation resulting in more centers. Consequently, less reduction rate. The variability in data of these days is high thus the reduction is less than other days.

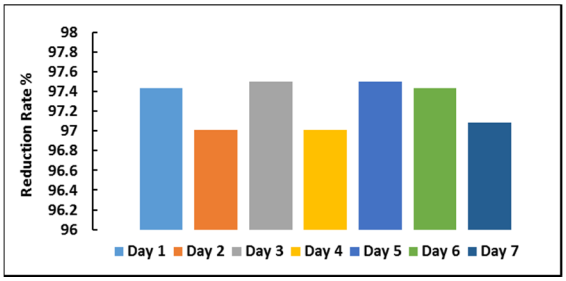



Fig. 5. Reduction rate

Fig. 6 shows, the accuracy of raw and representative data for temperature and humidity sensor using mean measure. The accuracy by mean measure is obtained by the following formula:

$$MD = M(C_i) - M(V_i) \quad (2)$$

Where: MD is the mean difference; M is the mean; $C_i$ is the number of centers resulting from clustering; $V_i$ is the total number of vectors in the input block. Based on "(2)", the mean difference for temperature sensor shown in Fig. 6(a) is **0.09.** As for the humidity sensor shown in Fig. 6(b) the mean difference is **0.07**. These results provide evidence to high accuracy, where the small difference in mean implies that the location of data distribution was not changed enormously, also, the data lost during processing is very little.

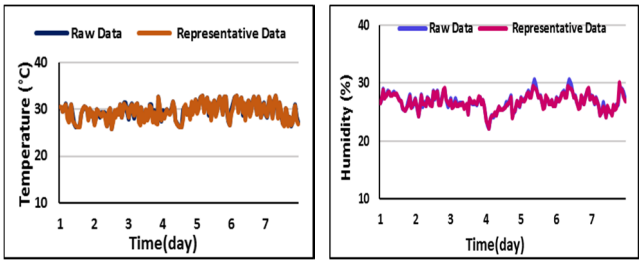



Fig. 6. Accuracy using mean measure: a) Temperature, b) Humidity.

The state is slightly different regarding the light sensor shown in Fig. 7(a). Compared with the other sensors, the mean difference for the light sensor is relatively high (2.98). Since the variability in data of this sensor is high as compared with the other sensors. Thus, the accuracy is relatively less than the other sensors. As for sound sensors shown in Fig. 7(b) the difference in the mean is **0.01.** This result indicates a high accuracy i.e. location of data distribution was not changed immensely, also, the data lost during processing is very little.

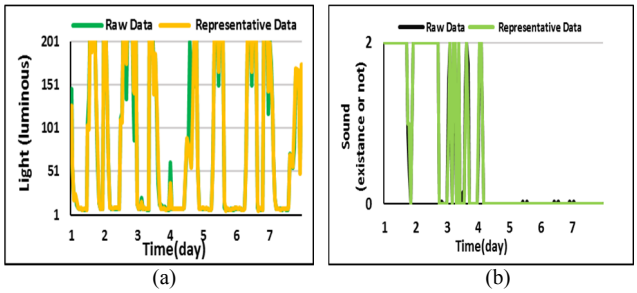
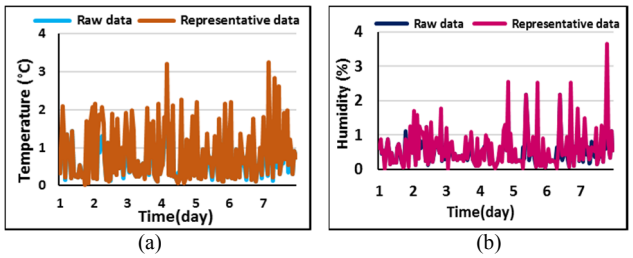



Fig. 7. Accuracy using mean measure: a) Light, b) Sound.

Fig. 8 shows the accuracy of raw and representative data for temperature and humidity sensor using a standard deviation measure. The accuracy by standard deviation measure is computed the following formula:

$$STD = AvgSd(C_i) - AvgSd(V_i) \quad (3)$$

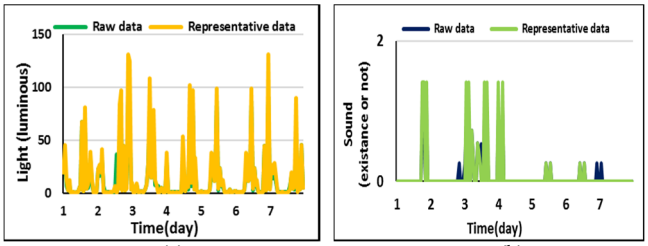
Where: STD is the standard deviation difference; AvgSd is the average standard deviation; $C_i$ is the number of centers resulting from clustering; Vi is the total number of vectors in the input block. According to "(3)", the average difference in the standard deviation of temperature sensor shown in Fig. 8(a) is **0.19.** As for the humidity sensor shown in Fig. 8(b) the average difference in standard deviation is **0.11**. These results give an indicator of high accuracy. Since the small standard deviation implies that the data points are near their mean and the dispersion of data is little, and therefore the accuracy is high.




(a) (b)

Fig. 8. Accuracy using standard deviation measure: a) Temperature, b) Humidity.

Regarding the light sensor shown in Fig. 9(a), the average difference in the standard deviation is relatively high (3.51) and the accuracy is relatively less than the other sensors. Since the data variability for this sensor is high and occurs in convergent intervals. Concerning the sound sensors illustrated in Fig. 9(b) the average difference in standard deviation is **0.03**. As previously mentioned, the small standard deviation implies that the data points are near their mean and the dispersion of data is little, and therefore the accuracy is high.




(a) (b)

Fig. 9. Accuracy using standard deviation measure: a) Temperature, b) Humidity.

We have compared our proposed approach with the works presented in [4, 20]. The comparison is performed based on the reduction rate achieved regardless of the data reduction technique used. According to the results in Fig. 5, it is noted that the proposed approach achieved a data reduction percentage in the range 96-97% in comparison with [4], which achieved a reduction rate percentage in the range 43.25-85.13% and [20] which achieved a reduction rate up of 95%. Comparison results confirm the proficiency of the proposed approach in reducing the streamed IoT data transmitted to the cloud**.**

## V. CONCLUSION

In this work, a testbed system is built based on fog computing concepts to reduce the stream IoT data transmitted to the cloud. For the first time, the subtractive clustering algorithm is employed for stream data clustering with high proficiency. Subtractive clustering is suitable for stream data since it does not require a predefined number of centers. It finds important points in stream data based on congestion of value to each other. Real data validate the result of the model. A high reduction rate is achieved and high accuracy through visualizing data before and after reduction. As a future work, deploying service at cloud center and fog node to compute the response time at fog and cloud level

## REFERENCES


[1] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, et al., "All one needs to know about fog computing and related edge computing paradigms: A complete survey," Journal of Systems Architecture, vol. 98, pp. 289-330, 2019.

[2] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog computing: Towards minimizing delay in the internet of things," in 2017 IEEE international conference on edge computing (EDGE), 2017, pp. 17-24.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE internet of things journal, vol. 3, pp. 637-646, 2016.

[4] W. M. Ismael, M. Gao, A. A. Al-Shargabi, and A. Zahary, "An In-Networking Double-Layered Data Reduction for Internet of Things (IoT)," Sensors, vol. 19, p. 795, 2019.

[5] T. Yu, X. Wang, and A. Shami, "A novel fog computing enabled temporal data reduction scheme in iot systems," in GLOBECOM 2017-2017 IEEE Global Communications Conference, 2017, pp. 1-5.

[6] J. Han, M. Kamber, and J. Pei, "Data mining concepts and techniques third edition," The Morgan Kaufmann Series in Data Management Systems, pp. 83-124, 2011.

[7] S. Ding, F. Wu, J. Qian, H. Jia, and F. Jin, "Research on data stream clustering algorithms," Artificial Intelligence Review, vol. 43, pp. 593-600, 2015.

[8] K. Bataineh, M. Naji, and M. Saqer, "A Comparison Study between Various Fuzzy Clustering Algorithms," Jordan Journal of Mechanical & Industrial Engineering, vol. 5, 2011.

[9] J. Rantala and H. Koivisto, "Optimised subtractive clustering for neuro-fuzzy models," in 3rd WSEAS international conference on fuzzy sets and fuzzy systems, 2002.

[10] K. Demirli, S. Cheng, and P. Muthukumaran, "Subtractive clustering based modeling of job sequencing with parametric search," Fuzzy Sets and Systems, vol. 137, pp. 235-270, 2003.

[11] S. M. Berneti, "Design of fuzzy subtractive clustering model using particle swarm optimization for the permeability prediction of the reservoir," International Journal of Computer Applications, vol. 29, pp. 33-37, 2011.

[12] L. T. Ngo and B. H. Pham, "A type-2 fuzzy subtractive clustering algorithm," in Mechanical Engineering and Technology, ed: Springer, 2012, pp. 395-402.

[13] B. Alturki, S. Reiff-Marganiec, and C. Perera, "A hybrid approach for data analytics for internet of things," in Proceedings of the Seventh International Conference on the Internet of Things, 2017, pp. 1-8.

[14] D. Borthakur, H. Dubey, N. Constant, L. Mahler, and K. Mankodiya, "Smart fog: Fog computing framework for unsupervised clustering analytics in wearable internet of things," in 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP), 2017, pp. 472-476.

[15] H. Djelouat, H. Baali, A. Amira, and F. Bensaali, "IoT based compressive sensing for ECG monitoring," in 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2017, pp. 183-189.

[16] K. Hossain and S. Roy, "A Data Compression and Storage Optimization Framework for IoT Sensor Data in Cloud Storage," in 2018 21st International Conference of Computer and Information Technology (ICCIT), 2018, pp. 1-6.

[17] A. K. M. Al-Qurabat, C. Abou Jaoude, and A. K. Idrees, "Two tier data reduction technique for reducing data transmission in IoT sensors," in 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), 2019, pp. 168-173.

[18] S. García, J. Luengo, and F. Herrera, Data preprocessing in data mining: Springer, 2015.

[19] K. Hammouda and F. Karray, "A comparative study of data clustering techniques," University of Waterloo, Ontario, Canada, vol. 1, 2000.

[20] Y. Fathy, P. Barnaghi, and R. Tafazolli, "An adaptive method for data reduction in the internet of things," in 2018 IEEE 4th World Forum on Internet of things (WF-IoT), 2018, pp. 729-735.