IJICTR
International Journal of Information & Communication Technology Research

Volume 16- Number 1 – 2024 (11 -19)

Energy Efficient Distributed Anomaly Detection using Semi-Supervised Models in IoT

Mostafa Shabani
Department of Computer Engineering and Information Technology
Shiraz University of Technology
Shiraz, Iran
mostafashabanigh@gmail.com

Omid Bushehrian*
Department of Computer Engineering and Information Technology
Shiraz University of Technology
Shiraz, Iran
bushehrian@sutech.ac.ir

Received: 10 August 2023 – Revised: 16 October February 2023 - Accepted: 25 December 2023

Abstract—**Utilizing IoT technologies for monitoring large-scale smart facilities such as power, water and gas distribution networks has been the subject of many studies recently. The aim is to detect anomalous events in the network due to elements' failure, bad designs, attacks or abuses of the network and alert the network operators in a timely manner. As the centralized cloud-based approaches are impractical in time-critical and real-time anomaly detection applications due to 1) high sensor-to-cloud transmission latency 2) high communication cost and 3) high energy consumption at the sensor nodes, the distributed anomaly detection methods based on Deep Neural Networks (DNN) have been applied in past studies vastly. In these methods, in order to detect anomalies in real-time, copies of the anomaly detection model are placed at the sensor nodes (rather than placing one at the cloud node) reducing the sensor-to-cloud transmissions significantly. Nevertheless, new normal samples collected at the sensor nodes still need to be transmitted to the cloud node at predefined intervals to re-train the distributed anomaly detection DNNs. In order to minimize these sensor-to-cloud transmissions during the retraining process, in this paper, two well-known lossless coding algorithms: Huffman Coding and Arithmetic Coding were studied and it was observed that the Huffman and Arithmetic Coding were able to reduce the transmission traffic up to 50% and 75% respectively using two IoT benchmark datasets of pipeline measurements. Besides, the Huffman Coding shown to be computationally feasible on resource limited sensors and resulted in up to 10% saving in energy consumption on each sensor resulting in longer network longevity. Moreover, the experimental results showed that the auto-encoder DNN could outperform the one-class SVM in the iterative distributed anomaly detection method.**

Keyword: distributed anomaly detection, auto-encoder, SVM, coding algorithm, IoT.

Article type: Research Article

Publisher: ICT Research Institute

* Corresponding Author

I. Introduction

Utilizing the IoT technologies for monitoring the large-scale smart city facilities such as intelligent transportation systems, utility distribution networks (power, water and gas distribution networks), waste collection and sewage collection networks has been the subject of many studies recently [1-3]. The aim is to detect anomalous events and incidents such as faulty elements, bad designs, attacks or abuses of the network and alert the operators in a timely manner. For monitoring the smart cities' infrastructures such as pipelines, these facilities are equipped with IoT devices and wireless sensors in order to measure local metrics as the raw data and process and transmit them to the edge of the network and subsequently to the cloud for further analysis [2]. Unexpected and unusual events in the environment such as element failures, attacks or abuses may cause measurements that lie far from the normal pattern of data and are called anomalies [4]. In other words, anomalies are unusual observations that differ from the majority of the data and anomaly detection is the process of identifying and reporting anomalous patterns [2]. There are three approaches in anomaly detection: (1) supervised; which requires labelled dataset; labels are either 'normal' or 'abnormal' (2) Unsupervised; in which labels are not needed and statistical methods are applied to spot outlying parts of data as anomalies and (3) Semi-supervised; where training data needs to be labelled, however only the normal data is used for the training [2]. Auto-encoder neural networks and one-class Support Vector Machines(SVM) are used in this approach [5]. Semi-supervised methods have shown very promising results in IoT applications and we have used them in this study as well.

With respect to the computation infrastructure, the anomaly detection task could be performed either at the cloud node (centralized architecture) or at the IoT sensors (distributed architecture). The former, in which the anomaly detection model is placed at the cloud node, involves high sensor-to-cloud transmission latency which is not acceptable particularly in time-critical and real-time IoT applications. Moreover, due to continuous transmission of sensor measurements to the cloud over the wide area network, the cloud-based model incurs high communication costs. In contrast, in the distributed architecture, the copies of Deep Neural Network (DNN) anomaly detection model are placed at the IoT sensors, to reduce the transmission costs and also to achieve desirable response time in anomaly detection process. However, there are two important challenges in the distributed anomaly detection models that needs to be addressed: (1) Training the anomaly detection model with the normal data could not be done on the sensor nodes due to the limited computational capacity on sensors and (2) the initial training dataset is very small and limited. To address the first challenge, the training process is performed on the cloud node and the model parameters are provided to the sensor nodes; however as mentioned in the second challenge, due to limited number of normal samples at the beginning, this process is repeated as new normal samples are detected by the model copies at sensors [5]. In this iterative training process, the new normal data samples collected at the sensors are transmitted to the cloud node to be used for the next training round. Here, the third challenge arises: (3) minimizing the incurred cost of transmitting the normal data samples from sensors to the cloud node for retraining purpose. This challenge is addressed in this paper. It is notable that the distributed and iterative anomaly detection model explained above is superior to the cloud-based model as it significantly lowers the anomaly detection latency due to the local DNNs replicated over the IoT sensors. As the DNN is trained at the cloud-node and not at the sensors, no extra operational and maintenance cost is needed in the distributed model. Moreover this model reduces the overall sensor-to-cloud transmission costs due to the fact that only the normal measurements are sent to the cloud at retraining intervals.

Two further reducing the sensor-to-cloud transmissions, in this paper the effect of applying two well-known lossless coding algorithms: Huffman Coding (HC) and Arithmetic Coding (AC) [6] on minimizing the communication traffic between sensors and cloud node are studied and an energy efficient distributed anomaly detection model is proposed. The contributions of this paper are summarized as follows:

- Augmenting the previous distributed anomaly detection models by adding a coding module to achieve energy efficiency.
- Comparing two well-known coding schemes based on their communication and computation costs in the context of IoT applications in processing the pipeline sensory measurements.
- Analyzing the impact of deploying two coding-schemes on the sensors from the energy consumption point of view.
- Studying the effect of two important semi-supervised learning models: auto-encoder DNN and one-class SVM on the overall accuracy of the distributed anomaly detection model.
- Conducting extensive experiments using two real-world IoT datasets in the area of gas industry.

The rest of this paper is organized as follows: section II presents the related works; the proposed model and coding schemes are explained in section III; the evaluation method, the experimental results and a discussion on the results are presented in section IV; and finally section V concludes the paper.

II. Related Works

The previous studies in the area of anomaly detection in IoT systems are categorized into unsupervised, supervised and semi-supervised methods [2, 7]: In [8] an unsupervised and scale-able K Nearest Neighbor (KNN) based method for anomaly detection in WSN was proposed to protect the network from faults and attacks. In this study, to cope with the lazy learning problem of the conventional KNN

algorithms a new hyper-grid based KNN method was presented to be used in online anomaly detection process. In [9] a distributed anomaly detection algorithm based on isolation forests is proposed for WSNs. Here a global detector model is built using the local detectors propagated by the neighboring sensors. This method applies the lightweight statistical isolation trees to detect anomalies rather than non-linear powerful methods such as neural networks. In [10] the application of hierarchical anomaly detection in detecting anomalous incidents in water distribution networks (WDN) has been studied. First, by using an unsupervised ellipsoidal clustering algorithm the model of sensory data is created and subsequently the outlying clusters are detected using a distance-based method. The performance of the algorithms is very dependent on the selected window size of the clustering algorithm that should be chosen carefully. Moreover, no energy saving mechanism has been presented in this study.

With respect to the supervised anomaly detection methods, in [11] a supervised time-series anomaly detection method based on Long Short Term Memory (LSTM) neural network has been proposed. LSTM networks are a variant of Recurrent Neural Networks (RNN) that are very effective in predicting future values based on past history of the data and hence they are useful in anomaly detection in the time-series data such as Vehicular Traffic Flow data. LSTM models are known as effective replacement for semi-supervised models such as auto-encoders in detecting anomalies. They are trained with time-series of normal measurements and then are used to detect outlying patterns. However, energy efficient transmission of normal subsequences to the cloud node to be used for training next models still remains a challenge. In [12] a Federated Learning (FR) based model is proposed to create a neural network anomaly detection model. In FR, each edge node creates its own local neural network model representing the local patterns of data and then transmits the model parameters to the server node to be aggregated with the parameters received from other edge nodes in order to create a combined model. The newly created consolidated model then is transferred back to the edge nodes for the next round. The downside of applying FR in the IoT systems is the incurred computational cost of training the neural network on the resource-limited IoT sensors.

In semi-supervised anomaly detection methods, opposed to the supervised methods, a model is trained using only the normal data samples and then is used to spot outlying samples. Regarding the semi-supervised anomaly detection category, in [13] a one-class Support Vector Machine (SVM) is proposed to detect outliers. Here a hyper-ellipsoid with minimum effective radius is fit around the normal data. In [14] to attain a real-time, accurate and also lightweight anomaly detection mechanism in WSNs, an online distributed method based on ellipsoidal one-class SVM has been proposed. They also considered the spatial-temporal correlations of data and keep their model updated to reflect the changes in the normal behavior of data over time. While one-class SVM is a means to detect anomalies, in previous studies no comparison has been conducted with other semi-supervised models such as auto-encoders. In [5] an iterative and distributed anomaly detection method using auto-encoder is presented. The initial auto-encoder model is built at the cloud node using a small number of training normal data samples. This initial model then is transmitted to the sensor nodes to be used for anomaly detection in the first round. The newly detected anomalous and normal data at the sensor nodes then are transmitted back to the cloud to be added to the previous dataset. At the next round the cloud node re-train the model using the new (larger) dataset and the above mentioned process repeats at the successive rounds. The advantage of this method is that the training overhead is not posed on the sensors and is done at the cloud node, however, transmitting the data samples from sensors to the cloud node poses high communication cost on the sensor nodes. In [15] to cope with the noisy training data and also capturing the spatial-temporal correlations in the normal data a deep learning-based anomaly detection algorithm was proposed. By applying a proper regularization method in a deep convolutional auto-encoder model, the first challenge is addressed. To address the second challenge a combination of linear and non-linear time-series prediction models has been applied. Although the model has shown promising results, it was not designed to be executed on a distributed computational infrastructure.

Following the previous studies in the area of semi-supervised anomaly detection, in this paper the distributed iterative methods using auto-encoders presented previously are augmented by adding a coding module to the sensor side in order to reduce the communication cost between sensors and the cloud node. In addition to the energy efficiency, we also compared the performance of auto-encoder models with one-class SVM in this framework when working on datasets of pipeline measurements in Gas Industry. A comparison between previous studies is presented in Table 1.

TABLE I. COMPARATIVE STUDY OF RELATED WORKS

Paper	Approach	method	Objective	main Differences with this study
[8]	Un-supervised	Hyper-grid KNN	Addressing the lazy learning problem of KNN	Non-iterative learning scheme
[9]	Un-supervised	Isolation Forest	Lightweight statistical method	No energy saving mechanism
[10]	Un-supervised	Hyper-ellipsoidal clustering	Scale-able outlier detection method	No energy saving mechanism
[11]	supervised	LSTM neural networks	measuring the distance of predicted values from the actual values in the time-series	No energy saving mechanism

[12]	supervised	Federated Learning	Anomaly detection while keeping the privacy of IoT data	No energy saving mechanism, High computation cost of training phase on sensor nodes
[13] and [14]	Semi-supervised	One-class SVM	Online, accurate and lightweight method	Non-iterative learning scheme ,No comparative study with auto-encoders
[5]	Semi-supervised	Auto-encoder	Iterative model of learning	No energy saving mechanism
[15]	Semi-supervised	Deep convolutional auto-encoder model	to cope with the noisy training data, capturing the spatial-temporal correlations in the normal data	Not designed for distributed computing

III. Energy Efficient Distributed Anomaly Detection

In this section first, the two important coding schemes are explained and subsequently the enhanced energy efficient iterative learning framework for IoT environments is presented in details.

A. *Coding Schemes*

Two important lossless coding schemes namely Huffman Coding (HC) and Arithmetic Coding (AC) [6] were studied in this paper. The former creates codes for symbols in the text based on the symbol frequencies. The idea is to have shorter codes for the symbols with higher occurrence in the text and vice versa. The concept of HC is illustrated in Fig.1 left where the input text "test" is coded with binary "101001" using a particular prefix tree called Huffman Tree. The HC algorithm first creates the tree where the leaves of the tree hold the symbols and any path from the root to a leaf determines the code for the respective symbol. The tree is created such that the symbols with lower frequencies are placed in deeper leaves and vice versa. In contrast to the HC that builds the code table first and then replaces the symbols in the text with the respective codes from the code table, in AC the entire source text is assigned a code arrived at by a rather complicated process. Methods in AC vary but they all have specific things in common: the source text is assigned a sub-interval from [0,1) that represents the source text. Afterwards, a fraction r in that sub-interval is chosen as the source code. Fraction r could be either decimal or binary. The larger the calculated sub-interval is, the fewer decimal places fraction r will have resulting in shorter code for the source text. Due to many multiplications of fractional numbers needed when coding the long source texts in AC, implementing AC in practice requires that a precision parameter P be provided to the algorithm. This precision parameter determines the max number of digits in the generated fractional code. Different implementations of AC vary in how to choose r and P. The concept of AC is illustrated in Fig. 1 right where the text "test" is coded with binary expansion "0.0101101". As shown in this figure (step by step from top to bottom), the symbol frequencies are used to partition the range [0,1) and to assign a sub-interval to each symbol whose length is proportional to its frequency. Afterwards, the whole text is scanned symbol by symbol from left to right and a sub-interval in the current interval is chosen corresponding to the current symbol. Finally we arrive at the sub-interval [0.01011, 0.010111] representing the whole text. Note that in the final sub-interval the boundaries are binary fractional numbers. A representative number within the final sub-interval is chosen as the code.

B. *Proposed Model*

The proposed model is based on the iterative distributed anomaly detection model presented in [5] that works as follows: An anomaly detection model is trained at the cloud node with available normal dataset. The objective is to train the auto-encoder model to reconstruct the normal data samples (model of the normal data). Copies of this model then are distributed among sensor nodes. Sensor nodes apply their model to discriminate between normal and anomalous measurements over predefined intervals. New normal data collected at the sensor nodes are transmitted to the cloud node to be added to the normal dataset. The process repeats from step 1. We have augmented the abovementioned model by adding a coding module to be used at step 3. The main function of the coding module is to compress the normal data so that the sensor-to-cloud transmissions at step 3 are lowered as much as possible. By reducing the transmission volume, not only is the communication costs reduced drastically but also the energy consumption at the sensor nodes is saved. Two important lossless coding algorithms: Huffman Coding (HC) and Arithmetic Coding (AC) were studied in this paper from the compression rate and computation cost perspectives (the results are reported in the Evaluation section). The architecture of the distributed anomaly detection with coding module is illustrated in Fig. 2. We also studied the effect of replacing the auto-encoder with one-class SVM as the anomaly detector models. As shown in Fig.2 right, the communication between sensors and the cloud node which involves both model parameters (published by the cloud node) and the normal data samples (published by the sensor nodes) takes place via a message broker over MQTT protocol which is usual in IoT applications. As presented in [5] (where auto-encoder is applied as the anomaly detector model), to discriminate between normal and anomalous data vectors at sensor s, vector x(s) is input to the auto-encoder to obtain output vector $\hat{x}(s)$. Then the deviation $r_x(s)$ is computed:

$$r_x(s) = x(s) - \hat{x}(s) \quad (1)$$

The label of x(s) is determined based on the distance between $r_x(s)$ and the mean of r(s) values for the normal data vectors:

$$Label_x(s) = \begin{cases} Normal: |r_x(s) - \mu| \leq p\sigma \\ Anomaly: otherwise \end{cases} \quad (2)$$

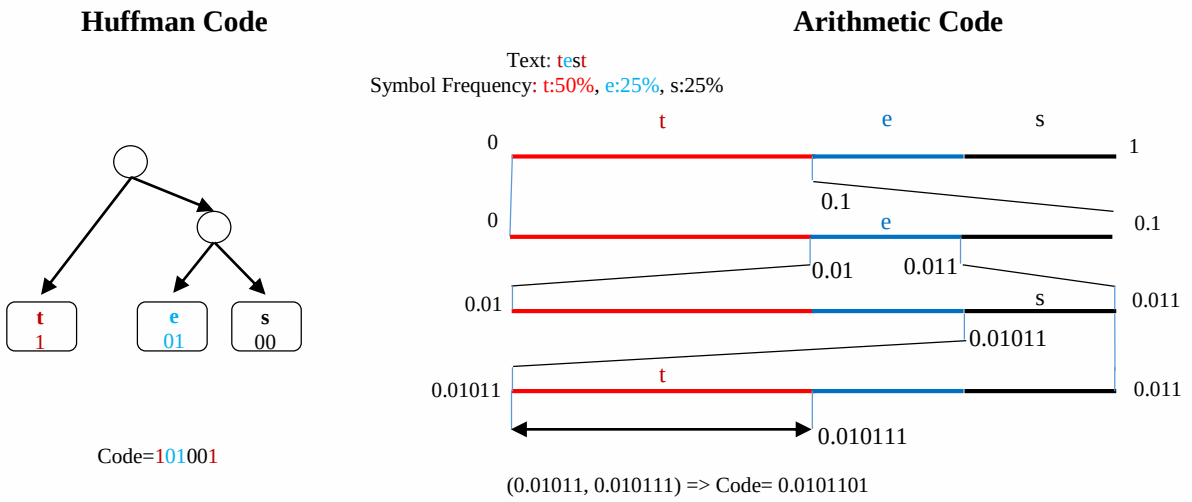

Figure 1. The coding schemes: (left) HC and (right) AC

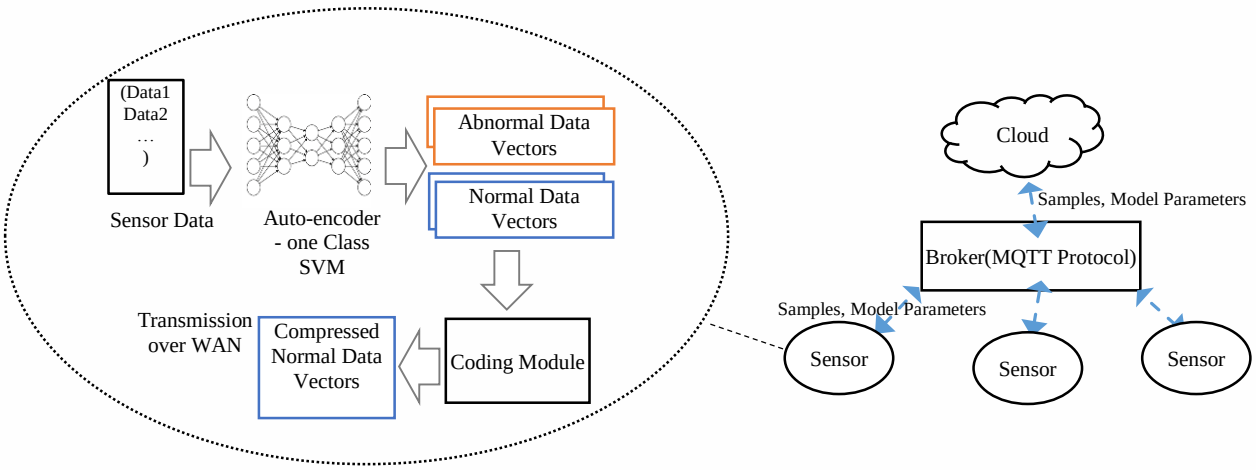

Figure 2. The distributed anomaly detection Architecture

Where μ and σ are the mean and mean deviation of r(s) values for all normal data vectors respectively and ρ is a constant parameter. The values of μ and σ are computed by the cloud node in successive intervals. The pseudo-code of the sensor and cloud algorithms are listed below in Listing 1 and 2 respectively.

The cloud node iteratively updates the auto-encoder model with newly obtained normal samples from the sensors. Sensors receive the updated model to be used for next anomaly detection step. In the case of one-class SVM, the hyper-sphere fitting method which is a special form of the hyper-ellipsoidal method has been applied [13][14].

While True:
Obtain sensor readings x(s)
Use auto-encoder to obtain $\hat{x}(s)$
Calculate residual $r(s) = x - \hat{x}$;
Label data vector x;
Compress and send x(s) and r(s) to the cloud;
Update model parameters (W, b, μ, σ) received from cloud;

Figure 3. The sensor procedure

```
While True:
    Receive x(s) from all sensor nodes;
    Store x(s) in the training dataset;
    If all sensor nodes data are received:
        Retrain auto-encoder with the updated training dataset;
        Recalculate μ, σ;
        Send updated parameters (W, b, μ, σ) to all sensors;
```

Figure 4. The cloud procedure

In the hyper-ellipsoidal method (the general problem) a hyper-ellipsoid with minimum effective radius is fit around the majority of data vectors centered at the origin. This is formulated as an optimization problem as follows [13]:

$$\text{Minimize:} \quad L(R,\xi i) = R^2 + \frac{1}{vn}\sum_{i=1}^{n}\xi i \qquad (3)$$

$$\text{Subject to:} \quad x_i \sum^{-1} x_i^T \le R^2 + \xi i\,, \xi i \ge 0, i = 1..n$$

Where n is the number of sample vectors in dataset, xi is the i^{th} sample vector, R is the effective radius of the hyper-ellipsoid, ξi is the slack factor allowing xi to reside outside the hyper-ellipsoid for a given R, v is the regularization parameter in range (0,1) and $\sum^{-1}$ is the inverse of the samples' covariance matrix. By replacing the covariance matrix $\sum$ with the unit matrix, the problem is reduced to the hyper-sphere based scheme. By solving this minimization problem, the effective minimum value for R will be obtained for which the resulting hyper-ellipsoid (or hyper-sphere) covers the majority of samples. To decide for a given sample z, the distance of z from the center of hyper-ellipsoid is computed; if the distance is greater than R, z will be classified as an anomalous sample.

IV. Evaluation

The objective of the evaluation has been to study the effect of applying HC and AC encoding algorithms in reducing the transmission traffic between sensors and the cloud node in the proposed architecture. Moreover, the effect of replacing the auto-encoder with one-class SVM as the anomaly detector model has been evaluated. To do the experiments a test-bed consisting of eight sensor nodes developed using Python communicating over MQTT using the Mosquitto broker [16] were used. To constrain the sensor node resources (CPU and Memory) as well as having an isolated runtime environment, each sensor was executed as a Docker [17] container. The HC and AC coding schemes were implemented using Numpy [18] and Decimal [19] libraries. The auto-encoder model is a fully connected neural network that uses Adam optimizer, Mean Squared Error Loss function (MSELoss) and Sigmoid activation function. For the ECG5000 dataset, the auto-encoder model consists of 7 layers and 140 input-output dimensions, for Gas dataset, which is a smaller dataset, auto-encoder model has 5 layers with 16 input-output dimensions. The learning rate of the model in both cases has been set to 0.0001. The one-class SVM model was implemented using the sklearn.svm.OneClassSVM python class [20]. This implementation of one-class SVM allowed us to make the classification either by means of the standard threshold or a customized threshold. To obtain the best accuracy we chose to use the customized threshold which is controlled by a percentile hyper-parameter s.

A. DataSets

In order to do the experiments two datasets were used: (1) ECG5000 dataset [21] consisting of 140 attributes and 5000 samples collected during 20 hours of sensory measurements and (2) a dataset consisting measurements from Gas pipelines obtained from [22]. Both datasets contain normal and anomalous samples. The normal samples were used to train the auto-encoder or one-class SVM models.

B. Results

As explained earlier, the distributed anomaly detection model works iteratively and in each iteration, sensors perform the anomaly detection process using the model obtained from the previous iteration from the cloud node. Afterwards, the sensors apply their current model to discriminate between normal and abnormal data and transmit the normal data to the cloud node for the next iteration training. The cloud node adds the new detected normal samples (by the sensor nodes) to its training dataset. However due to the fact that the accuracy of the detection is not perfect at sensors, particularly at the initial iterations, there are always some anomalous samples labeled as normal in this training dataset. Initially 2% of the normal samples are used to train the first model. This initial model then is re-trained gradually with more samples form the sensor nodes. Over the successive intervals, at each interval 4% of the data (normal and anomaly) are fed into the models at sensors to detect anomalies and hence the size of the cloud training dataset over the iterations is increased. In order to compare the accuracy of the auto-encoder and the one-class SVM, first we executed each model several times with different threshold values: p for the auto-encoder and s for the one-class SVM as explained in section III. The threshold value that resulted in the highest average F1-Score over iterations for each model (one-class SVM or auto-encoder) and each dataset was selected for further comparisons(s=72%,p=0.9). Here, F1-score is the average of Normal and Anomaly classes F1-scores. The comparison of average F1-score of auto-encoder and one-class SVM in two datasets over successive iterations is presented in Fig.3. It was observed that for the larger dataset (ECG5000) the auto-encoder model obviously outperformed the one-class SVM. In contrast to the deep neural network auto-encoder, the one-class SVM was unable to learn the complex patterns of normal data samples in a large dataset like ECG5000. Moreover, not only could not the one-class SVM outperform the auto-encoder in the large dataset, but also its trend of F1-score values over successive iterations was not increasing opposed to the auto-encoder model(in both datasets). The failure of one-class SVM in reaching higher detection accuracy as the number of training samples increase in the cloud, makes this model an improper choice for the iterative learning framework explained in this paper where few normal samples are available at early iterations. In the Gas dataset, the F1-score values of the one-class SVM is higher, however the trend is still not increasing. This deficiency of the one-class SVM in gradual anomaly detection is observed in Fig.4 where the ratios of

collected normal samples in the cloud node over iterations are depicted for the Gas dataset. As expected, this ratio had an obvious increasing trend when using the auto-encoder model; whereas the trend for the one-class SVM is not increasing and even it is the reverse. Regarding the reduction in the transmission traffic, the number of published characters over MQTT was compared for three methods namely, HC, AC and the baseline method and the results are shown in Fig. 5.

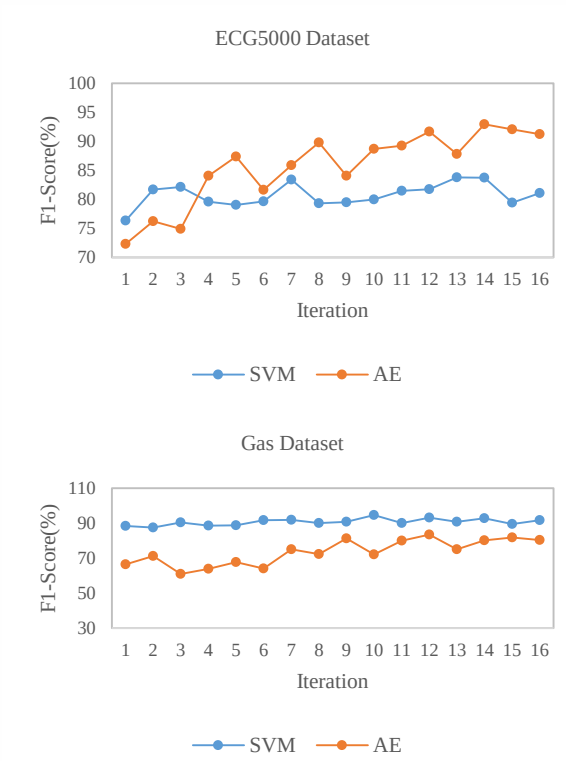

Figure 5. The F1-Score (Average of Normal and Anomaly classes) of auto-encoder and SVM models in two datasets.

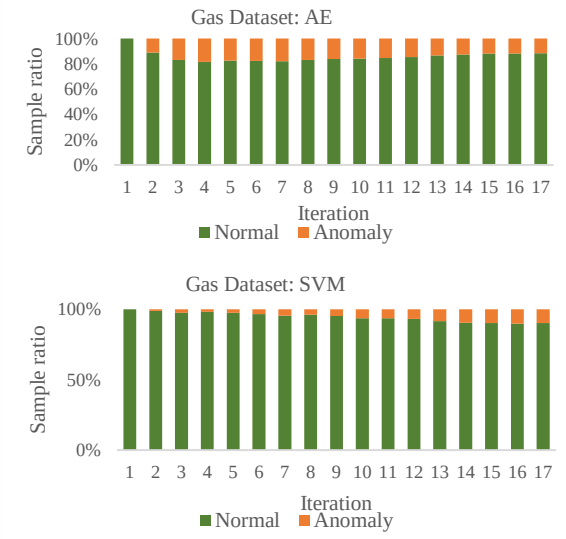

Figure 6. The percentage of normal data samples at the cloud node over successive iterations: auto-encoder (top) and one-class SVM(bottom).

The compression rate is also shown in Fig. 6. The AC encoding method outperformed the HC and the baseline algorithms in reducing the transmission traffic between sensor nodes and the cloud node and it showed a higher compression rate when applying on the sensory measurements in both datasets. As the frequency of symbols in the transmitted text is almost uniform in successive intervals, the compression rate of HC was steady as shown in Fig. 6.

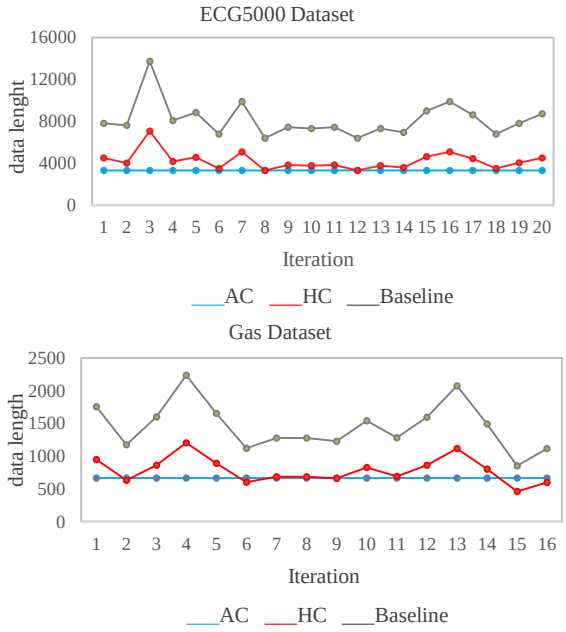

Figure 7. The number of published characters in successive intervals for two coding schemes in two datasets

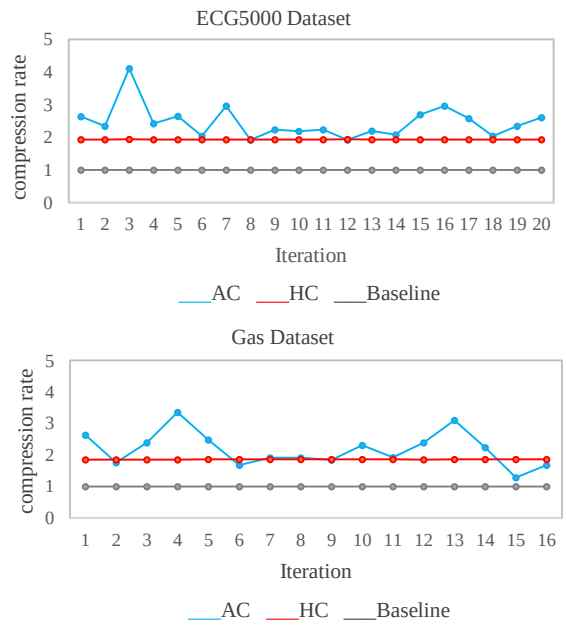

Figure 8. The compression rate in successive intervals for two coding schemes in two datasets

In order to compare the computation cost of AC and HC coding algorithms when used on sensors, they were executed within resource-limited containers for different input lengths. The results are shown in Fig. 7-top. As shown in this figure, the AC scheme had much higher computation cost in terms of the compression time due to its more complex algorithm as explained in section III. In contrast, the HC scheme scaled very well by increasing the input text length when working on the

sensory data. To compare the total energy consumption of the baseline and HC methods, the Cooja [23] simulator was used. Sensors of type "sky mote" were used to collect the energy consumption during the simulation time. The results are shown in Fig. 7-bottom. Due to limitations of the Cooja simulator implementing the AC method was not possible. Although we already knew that the computation cost of AC method is much higher than HC (see Fig. 7-top) and AC is not a proper coding scheme to be used on resource-limited sensors. Hence only the HC and baseline schemes were compared. As shown in this figure, by using the HC method, sensors could save energy compared to the baseline method due to much lower data transmission volume and little computation overhead posed by HC scheme. The amount of saving in energy rises for longer sequences of sensory measurements.

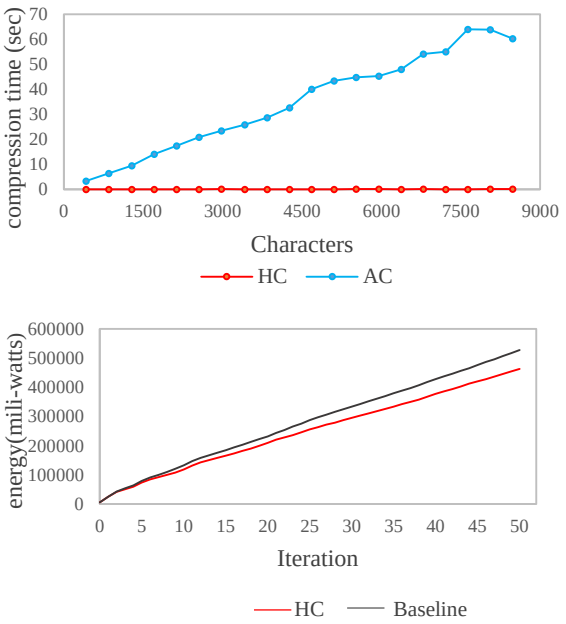

Figure 9. The comparison of: (top) the computation cost of the encoding schemes and (bottom) HC coding scheme and the Baseline methods in terms of the total energy consumption in sensors.

C. Discussion

Due to the larger training dataset at the cloud node in later iterations, the auto-encoder model is trained with more normal samples and hence not only does the accuracy of the model increase over time but also the proportion of normal samples at the cloud dataset increases due to the enhanced ability of the model in discriminating normal and anomalies. However this behavior was not observed when using the one-class SVM. The accuracy of one-class SVM was not rising with more samples as it seems that the hyper-sphere which is fit around the majority of samples remains the same in successive iterations even if new patterns of normal data emerge. This deficiency of the one-class SVM makes it improper choice for iterative learning frameworks. Moreover due to the fact that deep neural network auto-encoder model takes advantage of non-linearity and more complex structure compared to the one-class SVM it could outperform the one-class SVM in the larger dataset. It was observed (see Fig. 5 and Fig. 6) that the AC coding method outperformed the HC and the baseline algorithms in reducing the transmission traffic between sensor nodes and the cloud node due to its more complex coding algorithm. Despite the higher compression rate of the AC method, it showed poor performance in terms of execution time as shown in Fig. 7-left due to its higher algorithm time complexity compared to the HC coding when used on resource-constrained sensors. By using the HC method on sensors, not only could it reduce the communication cost up to 50% but also it caused saving in the sensor consumed energy (as shown in Fig. 7-right) due to less utilization of the sensor communication interface.

V. Conclusions

As the centralized cloud-based approaches are impractical in time-critical anomaly detection IoT applications due to large sensor-to-cloud transmission latencies, high communication costs and high energy consumption at the sensor nodes, the distributed anomaly detection methods based on DNNs are used as a replacement. In this paper the effectiveness of applying two well-known lossless coding algorithms, namely Huffman Coding (HC) and Arithmetic Coding (AC) in data transmission over IoT infrastructures was studied within the distributed and iterative anomaly detection framework. By using auto-encoder DNNs on two standard benchmarks, the experimental results showed that HC coding scheme not only reduces the size of published messaged up to 50% but also poses negligible computation cost on the sensors and hence could result in up to 10% energy saving when implemented on sensors compared to the baseline method. By replacing the auto-encoder model with one-class SVM in the iterative framework, the training process in the cloud was faster but a drop in the prediction F1-score particularly in larger datasets was observed. Moreover the one-class SVM was unable to learn more from samples collected in later iterations. As the future work, we aim to study the energy-effectiveness and accuracy of applying the federated learning in place of the centralized learning model over the Edge computing infrastructure.

Acknowledgment

This research was supported by Fars Province Gas Company.

References

[1] A.S. Syed, D. Sierra-Sosa, A. Kumar, A. Elmaghraby, "IoT in smart cities: a survey of technologies, practices and challenges", Smart Cities, 2021. 4(2): p. 429-475.

[2] V. Hodge and J. Austin, "A survey of outlier detection methodologies", Artificial intelligence review, 2004. 22(2): p. 85-126.

[3] E. Theodoridis, G. Mylonas and I. Chatzigiannakis, "Developing an iot smart city framework", in IISA 2013, 2013. IEEE.

[4] A. A. Cook, G. Mısırlı and Z. Fan, "Anomaly Detection for IoT Time-Series Data: A Survey," in *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6481-6494, July 2020, doi: 10.1109/JIOT.2019.2958185.

[5] T. Luo and S. G. Nagarajan, "Distributed Anomaly Detection Using Autoencoder Neural Networks in WSN for IoT," *2018 IEEE International Conference on Communications (ICC)*, Kansas City, MO, USA, 2018, pp. 1-6, doi: 10.1109/ICC.2018.8422402.

[6] P.D. Johnson Jr, G.A. Harris and D.C. Hankerson, Introduction to information theory and data compression, 2003: Chapman and Hall/CRC.

[7] A. Ayadi, O. Ghorbel, A. M. Obeid and M. Abid, "Outlier detection approaches for wireless sensor networks: A survey", Computer Networks, 2017. 129: p. 319-333.

[8] M. Xie, J. Hu, S. Han and H. -H. Chen, "Scalable Hypergrid k-NN-Based Online Anomaly Detection in Wireless Sensor Networks," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 8, pp. 1661-1670, Aug. 2013, doi: 10.1109/TPDS.2012.261.

[9] ZG. Ding, DJ. Du and MR. Fei, "An isolation principle based distributed anomaly detection method in wireless sensor networks", *Int. J. Autom. Comput.* **12**, 402–412, 2015, https://doi.org/10.1007/s11633-014-0847-9.

[10] S. Mirzaie, M.R. AvazAghaei and O. Bushehrian, "Anomaly Detection in Non-Stationary Water Distribution Grids Using Fog Computing Architecture", International Journal of Information and Communication Technology Research, 2021, 13(3): p. 12-23.

[11] W. Jia, R. M. Shukla and S. Sengupta, "Anomaly Detection using Supervised Learning and Multiple Statistical Methods," *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, Boca Raton, FL, USA, 2019, pp. 1291-1297, doi: 10.1109/ICMLA.2019.00211.

[12] S. Kim, H. Cai, C. Hua, P. Gu, W. Xu and J. Park, "Collaborative Anomaly Detection for Internet of Things based on Federated Learning," *2020 IEEE/CIC International Conference on Communications in China (ICCC)*, Chongqing, China, 2020, pp. 623-628, doi: 10.1109/ICCC49849.2020.9238913.

[13] S. Rajasegarar, C. Leckie and M. Palaniswami, "CESVM: Centered Hyperellipsoidal Support Vector Machine Based Anomaly Detection," *2008 IEEE International Conference on Communications*, Beijing, China, 2008, pp. 1610-1614, doi: 10.1109/ICC.2008.311.

[14] Y. Zhang, N. Meratnia, and P.J. Havinga, "Distributed online outlier detection in wireless sensor networks using ellipsoidal support vector machine", Ad hoc networks, 2013. 11(3): p. 1062-1074.

[15] Y. Zhang, Y. Chen, J. Wang, Z. Pan, "Unsupervised deep anomaly detection for multi-sensor time-series signals", https://doi.org/10.48550/arXiv.2017.12626 , 2021.

[16] Mosquitto Broker. [cited 2022 June 1, 2022]; Available from: https://mosquitto.org/.

[17] Docker. [cited 2022 22 Aug]; Available from: https://www.docker.com/.

[18] Numpy python library. [cited 2022 jul 25]; Available from: https://github.com/numpy/numpy.

[19] Decimal fixed point and floating point arithmetic. [cited 2022 jul 25]; Available from: https://docs.python.org/3/library/decimal.html.

[20] L. Buitinck et al. "API design for machine learning software: experiences from the scikit-learn project", https://doi.org/10.48550/arXiv.1309.0238, 2013.

[21] Y. Chen , E.K. "ECG5000" Dataset. n.d. June 1, 2022]; Available from: http://www.timeseriesclassification.com/description.php?Dataset=ECG5000.

[22] Pressure Sensors towards Pipeline Leakage Detection Dataset. [cited 2022 9 Aug]; Available from: https://zenodo.org/record/4769101#.YxNSQ3ZBwon.

[23] Cooja Simulator, Available from: https://anrg.usc.edu/contiki/index.php/Cooja_Simulator

Mostafa Shabani received his B.Sc. degree in Computer Science from Shamsipour Technical and Vocational College, Tehran, Iran, in 2018, and M.Sc. degree in Computer Network Engineering from Shiraz University of Technology, Shiraz, Iran, in 2023. His research interests are IoT, Anomaly Detection and Machine Learning.

Omid Bushehrian received his B.Sc. in Software Engineering from Amirkabir University of Technology (Tehran poly-techniques) in 2001. He received his M.Sc. and Ph.D. degrees from Iran University of Science and Technology (IUST) in Software Engineering in 2003 and 2008 respectively. He is currently an Associate Professor at Shiraz University of Technology working on different areas related to the Distributed Computing. His research interests include IoT, Application Migration to Cloud, Distributed and Large Scale Systems. He also has been working in telecom companies since 2008 as a Software Project Manager and Consultant.