Performance of Data Reduction Algorithms for Wireless Sensor Network (WSN) using Different Real-Time Datasets: Analysis Study

M. K. Hussein[1], Ion Marghescu[2]
Dept. Electronics, Telecommunication & Information Technology
University Politehnica of Bucharest
Bucharest, Romania

Nayef.A.M. Alduais[3]
Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia
Johor, Malaysia

***Abstract*—This paper investigates the effect of data reduction methods in the performance of Wireless Sensor Network (WSN) using a variety of real-time datasets. The simulation tests are carried out in MATLAB for several methods of reducing the quantity of sent data. These approaches are Data Reduction based - Neural Network Fitting (NNF), Neural Network Time Series (NNTS), Linear Regression with Multiple Variables (LRMV), Data Reduction based – "An Efficient Data Collection and Dissemination (EDCD2)" and Data Reduction based – Fast Independent Component Analysis (FICA). The selected algorithms NNF, NNST, EDCD2, LRMV, and FICA are evaluated using real-time datasets. The performance indicators included are energy consumption, data accuracy, and data reduction percentage. The research results show that the selected algorithm helps to reduce the amount of data transferred and consumed energy, but each algorithm performs differently depending on the dataset used.**

Keywords*—Data reduction algorithms; WSN; energy consumption; accuracy; neural network; independent component analysis

I. Introduction

In this paper, Wireless Sensor Network (WSN) is a network that collects data from spatially isolated sensors. Sensor nodes are used to monitor and record environmental variables, such as sound, pollution level, humidity, temperature, and wind, and then send the sensed data to the base station [1][2]. The sensor node in the WSNs application is powered by a battery with limited-service life [3]. Furthermore, sensor nodes with multivariable sensors can have an impact on battery life because the node must support additional data transmission, causing the battery to drain faster than a sensor node with a single sensor [4]. Therefore, many researchers have been proposed various approaches to reduce the amount of transmitted data at the sensor node level, which will help in prolonging the battery lifetime. For example, in WSN, the spatial and temporal correlation between the generated traffic can be used to reduce the energy consumption of continuous sensor data acquisition. Spatial-temporal correlation is used in dual prediction (DP) and data compression (DC) techniques to reduce the number of transmissions to save energy and bandwidth. In [5], the author has used these two technologies as part of a two-stage data reduction scheme. The DP technology reduces traffic between cluster nodes and cluster heads, while the DC scheme reduces traffic between cluster heads and sink nodes.

In [6], the author proposed a data-aware energy-saving technology. The essential correlation between continuous measurements of sensor nodes and the similarity of data trends between adjacent sensor nodes are used to reduce data transmissions. The forecast-based data collection framework reduces time data redundancy. "Autoregressive Integrated Moving Average (ARIMA)" model was used to predict data. The proposed model was implemented in the Cluster head (CH) node.

In [7], the author proposed a novel technique for secure data prediction in WSN by using a Time Series Trust Model (TSTM) based on the Toeplitz matrix and a trust-based autoregressive process (TAR). The author proposed an adaptive data reduction method (AM-DR) in [8]. AM-DR is based on a convex combination of two decoupled Least-Mean-Square (LMS) window filters of different widths for predicting the next readings at both the source and sink nodes.

In [9], the authors have evaluated the performance of several methods based on computational intelligence to decrease the amount of the payload of every packets sent from the sensor node to the base station. These approaches are data reduction based on "artificial neural networks (DR-ANN)"; independent component analysis (DR-ICA) and deep learning regression methods called DR-GDMLR".

In [10], two multivariate data reduction methods for adaptive thresholds were proposed a Principal Component Analysis Based (PCA-B) –and multiple linear regression Based (MLR-B). PCA-B is a multivariate data reduction method. It uses "Candid Covariance-free Incremental PCA (CCIPCA)" with an adaptive threshold and to reach a high reduction ratio the number of Principal Components (PCs) assigned to "1". Another method to decrease the amount of payload sensed data is named MLR-B, which it using multiple linear regression (MLR) model. The authors used an adaptive threshold to retrain the model. According to [10], after updating the reference parameters of the model, the size of the transmitted data is greater than or equal to the size of the payload data without being reduced. This means that the sensor node needs

more power during the update phase, than the required power during the phase of reduction. The article recommends a new indicator for evaluating the performance of data reduction models, which it considering the number of repeating of updating the parameters reference of the model. A novel simple scheme called "Adaptive Real-Time Payload Data Reduction Scheme (APRS)" is proposed in [4]. APRS purposes is to decrease the size of the transferred payload of the sensor nodes. Further details on approaches of data reduction for sensor nodes are defined in [11]. In this study, effect of data reduction approaches on WSN performance is investigated using a set of real-time datasets. Simulation tests are performed in MATLAB for different approaches to decrease the amount of transferred payload data. The selected algorithms NNF, NNST, EDCD2, LRMV, and FICA are evaluated using real-time data sets. The performance indicators included are energy consumption, data accuracy, and data reduction percentage.

The organization of the article is as follows: Section I presents the introduction and related work of this study and the main contributions. The selected data reduction algorithms are described in Section II. Section III explores the real-time datasets used in this study. Section IV describes the performance metrics used in this study to evaluate the algorithms. Section V presents the study simulation and results. Lastly, Section VI concludes the outcome of the study.

II. Selected Data Reduction Algorithms

A. Data Reduction based - Neural Network Fitting (NNF) Algorithm

The NNF model provided by MathWorks [12], helps in solving a data fitting problem using a two-layer feed-forward network. It helps in selecting the data, partitioning it into training, validation, and testing sets, defining the network architecture, and training the network.

In this section, the application of the NNF model to reduce the size of data transferred is described in detail. Fig. 1 represents the block diagram of WSN data reduction based on the NNF algorithm with a general structure. In the training phase, first select the sensor S1(t) with the highest correlation attribute as the input data of the NNF model and the other sensor features S2(t) and S3(t) as the output target of the NNF. The main objective in training NNF is to predict the values PS2(t) and PS3(t) from a single input sensor S1(t) during the reduction phase. As mentioned earlier, NNF is used to decrease the size of the transmitted data by the sensor node.

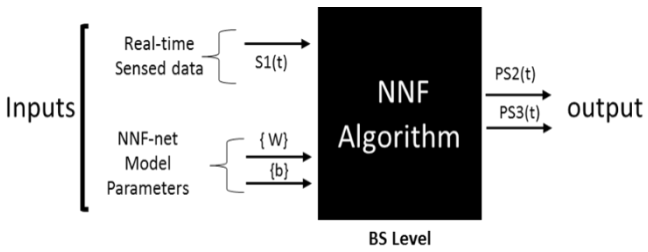

Fig. 1. General Block Diagram of NNF Algorithm.

The detailed description of the data reduction based NNF algorithm is stated by means of the following pseudocode.

NNF Model

Input: Inputs, targets
Output: Net // *NNF Model* **with**
Begin:
// Phase I *: Create a Fitting Network //*
Set Hidden Layer Size ←10
Set net ← **Call *FITNET*** (Hidden Layer Size)
Select InputOutput Pre-Post-Processing-Functions// *n=1, m=2*
Set net. Inputs{n}. process-Fcns←'removeconstantrows','mapminmax'
Set net.outputs{m}.process-Fcns←'removeconstantrows','mapminmax'
// Phase II *: Setup Division of Data for Training, Validation, Testing//*
Set TrainRatio, ValRatio and TestRatio ← {70,15,15}
Assign the training function // "Levenberg-Marquardt backpropagation
Select a Performance_Function// "
Set NETPERFORMFCN ← 'mse'; % Mean squared error
Phase III // Network -Train
Set [net,tr] ← *Call* ***TRAIN***(net,inputs,targets)
End

NNF algorithm

Input: S1(t), S2(t), S3(t) // Sensor value for S1 // real-time data
Output: **P**S2(t), PS3(t)
Begin:
Call *NNF Model*
For i=1 to M do // *M is the number of samples*
Send S1(i) → BS // Send vaule of the sensor 1 To BS
*// **At BS** //*
Estimate [$PS2(i), PS3(i)$] ← ***NNF****(S1(i))* // **Estimated data by NNF**
*// **Calculate error** // This step for check the performance of the algorithm*
Err2(i) ← ABS($PS2(i) - S2(i)$)
Err3(i) ← ABS($PS3(i) - S3(i)$)
End
End

B. Data Reduction based - Neural Network Time Series (NNTS) Algorithm

The prediction model NNTS provides by MathWorks [12]. NNTS is a type of dynamic filtering, that uses past-values of one or more-time series to predict future values. Dynamic neural networks containing tapped delay lines are used for nonlinear filtering and prediction.

This section describes in detail the NNTS algorithm used to reduce the amount of data transmitted. Fig. 2 represents the block diagram of WSN data reduction based on the NNTS algorithm with a general structure. In the training phase, first select the sensor S1(t) with a high correlation attribute as the input data of the NNTS model and the other sensor features S2(t) and S3(t) as the output target of NNTS. The main objective in training NNTS is to predict the values PS2(t) and PS3(t) from a single input sensor S1(t) during the reduction phase, where S1(t-1) and S1(t-2) are the last two received values of sensor S1(t). As mentioned earlier, NNTS is used to decrease the size of the transmitted data by the sensor node.

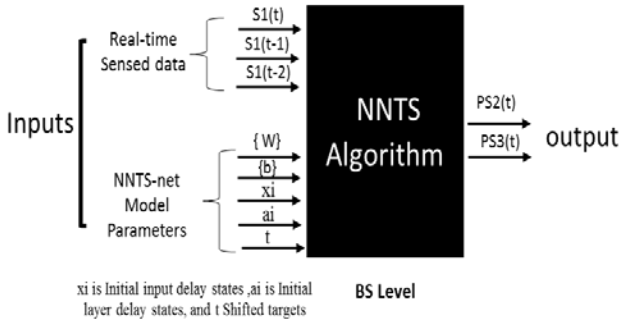

Fig. 2. General Block Diagram of NNTS Algorithm.

The detailed description of the data reduction based NNTS algorithm is given in the following pseudocode.

// Training NNTS Model//

Input: Inputs,Targets
Output: Net // *NNTS Model*
Begin:
// Phase I *: InputOutput TimeSeries Problem with a Time Delay Neural_ Network//*
Convert *data_to_ standard NNet cell array form using Tonndata_Fun*
Set X ← *Tonndata* (Inputs,false,false)
Set T ← Tonndata (Targets, false, false)
Create *a Time Delay Network*
Set Input_Delays ←1:2
Set Hidden_LayerSize ←10
Assign the trainingfunction //" Levenberg-Marquardt backpropagation"
TrainFcn ← 'trainlm'
Set net ← **Call *Timedelaynet****(InputDelays,HiddenLayerSize,TrainFcn)*
Select Input_Output Pre-Post-ProcessingFunctions//
Set net_inputs .process_Fcns←'removeconstantrows','mapminmax'
Set net_outputs. Process_Fcns←'removeconstantrows','mapminmax'
Prepare *TrainingData using Preparet_Fun*
Set [x,xi,ai,t] ← Preparets(net,X,T) // x is Shifted inputs, xi is Initial inputdelay states ,ai is Initial_layer_ delay_states, and t Shifted targets
// Phase II: Setup Division of Data for Training, Validation, Testing//
Set Train_Ratio, Val_Ratio and Test_Ratio ← {**70**, **15**, **15**}
Select a Performanc_ Function//
Set NET_PERFORMFCN ← 'mse'; % Mean squared error
Phase III // Train the Network
Set [net,tr] ← *Call* TRAINFUN (net,x,t,xi,ai)
End

NNTS algorithm

Input: S1(t), S2(t), S3(t) // Sensor values // real-time data
Output: **P**S2(t), PS3(t)
Begin:
Call *NNTS Model*
For i=1 to M do // *M is the number of samples*
Send S1(i) → BS // Send vaule of the sensor 1 To BS
// At BS //
Set *xi ← ([S1(i-1), S1(i-2)])// The recent two received values of the sensor 1*
Determine $PS2(i), PS3(i) \leftarrow$ ***NNTS****(S1(i*), xi,ai*) //* **Estimated data by NNTS**
// Calculate error *// This step for check the performance of the algorithm*
Err2(i) ← ABS($PS2(i) - S2(i)$)
Err3(i) ← ABS($PS3(i) - S3(i)$)
End for
End Algorithm

C. Data Reduction based – Linear Regression with Multiple Variables (LRMV) Algorithm

In statistics, linear regression is a linear approach to modeling the relationship between a scalar response and one or more explanatory variables (also referred to as dependent and independent variables). The theoretical concept of using linear regression with multiple variables was explained in detail by Ng, Andrew in [13].

In this section, the application of the LRMV algorithm to reduce the amount of data transferred is described in detail. Fig. 3 represents the block diagram of WSN data reduction based on the LRMV algorithm with a general structure, where the sensor S1(t) is assigned as the dependent variable of the LRMV model, and the other sensor features S2(t) and S3(t) are assigned as the predictor/independent variables of LRMV during the training phase. The aim of training LRMV is to predict the PS1(t) value from multiple sensors S2(t) and S3(t) during the reduction phase. The LRMV parameters are theta (θ), mean (mu), and standard deviation (SSDV). As mentioned earlier, LRMV the size of the transmitted data by the sensor node.

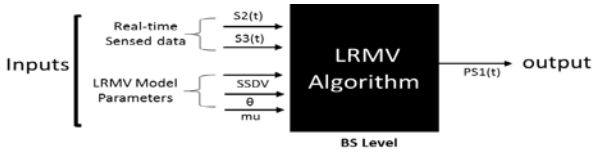

Fig. 3. General Block Diagram of LRMV Algorithm.

The detailed description of the data reduction based LRMV algorithm is stated in the following pseudocode.

// Training LRMV Model//

Input: Training data **// Sensed** data S1, S2, S3*// Features*
Output: theta - θ, Mean - mu and standard deviation - SSDV // *LRMV Model parameters*
Begin
// **Phase *I: Load Data and Initialize Variables*** //
Load Sensor dataset // *S1, S2, S3*
Set X ← *[S2 S3]*
Set Y← *S1//*
// Set Initialize Variables //
Set M ← *Number of training samples//*
Set D ← *Number of features//*
Initialize $\theta \leftarrow$ ZEROS(D+1,1); // Initialize thetas to zero.
Initialize Number_Itr ← N; // Set the number of repetitions for gradient descent.
Initialize $\alpha \leftarrow 0.5$; // *Set alpha Learning rate*
// **Phase II**: ***// Calculate Theta from Normal Equation****// data processing*
Set XNormEqn ← [ones(M,1) X]
Calculate thetaNormEqn ← **Call** *NormalEquation*(XNormEqn,Y)
//Phase III // *Feature Normalization//*
Normalizing Features for gradient descent
Calculate [X, mu, stddev] ← **Call** *featureNormalize***(**X**)**
Calculating β via gradient descent
$\theta \leftarrow$ **Call** *gradientDescent*(X, θ, Y, α, Number_Itr)
End

LRMV algorithm

Input: S1(t), S2(t), S3(t) // Sensor values // real-time data
Output: PS1(t)
Begin:
[θ, mu, stddev **] ←Call *LRMV Model***
For i=1 to M do // *M is the number of samples*
Send S2(i), S3(i) → *BS // Send vaule of the two sensors 2,3 To BS*
// At BS //
Set *X ← ([S1(i), S2(i)])// The recent received values of the sensor 2 and 3*
Determine $PS1(i) \leftarrow$ ***LRMV_Prediction_Fun*** *(X,* θ, mu, stddev*) // Estimated data by LRMV Prediction Fun*
*// **Calculate error** // This step for check the performance of the algorithm*
Err1(i) ← ABS($PS1(i) - S1(i)$)
End for
End

D. Data Reduction based –EDCD2 Algorithm

EDCD2 is a scheme to bring up-to-date measured data to the BS [14]. EDCD2 was used to decrease the number of transferred packets from nodes (multiple sensors). It should be noted that there are two versions of EDCD, EDCD1, and EDCD2 for sensor nodes with one and multiple sensors, respectively.

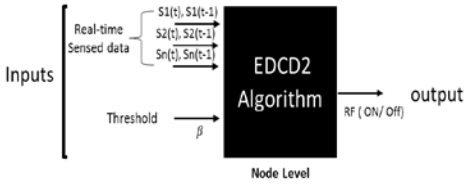

Fig. 4. General Block Diagram of EDCD2 Algorithm.

In this section, the application of the EDCD2 to reduce the size of data transferred is described in detail. Fig. 4 shows the block diagram of WSN data reduction based on the EDCD2 algorithm with a general structure. The basic idea of EDCD2 is to avoid transmitting the sensed data if the value of the relative difference between the currently sensed data S(t) and the last transmitted data S(t-1) is smaller than the threshold value β for all sensors of the same node, otherwise, the sensed data S(t) will be transmitted to the BS. The detailed description of the EDCD2 algorithm based on data reduction is given in the following pseudocode.

//EDCD2//

Inputs:
$S_i(t), S_i(t-1)$ for each sensor $S_i$and β.
Output: D_s
Begin:
For $i = 1{:}n$ Do // i=1, 2,…n ;
Set $S_i(t-1)\leftarrow$ last measuring value transmitted by the sensor S_i
Read: the sensor value (SV_i) at t time
Set$S_i(t) \leftarrow (SV_i)$
//Calculate the relative differences (R_f)
R_f=Abs $(S(t) - S(t-1))$ / $(S(t) + S(t-1)) \times 0.5)$
If $R_f > \beta$ Then
Set $SS_i \leftarrow 1$
Else: Set $SS_i \leftarrow 0$
End if
End For
// Recalculate the node data size (D_s)
Set $D_s \leftarrow 0$;
For $i = 1{:}n$ Do
$D_s = (D_s + (SV_i \times SS_i))$
End For
// The decision to send data
If $D_s = 0$ Then
RFtransmit (Off) // no update / no send
Else: RFtransmit (On) // update(send)
End If
End Algorithm

E. Data Reduction based – Fast Independent Component Analysis (FICA) Algorithm

Fast Independent Component Analysis (FICA) is an efficient and popular algorithm for independent component analysis developed by Aapo Hyvaerinen at Helsinki University of Technology. [15][16]. Like most FICA algorithms, FICA searches for an orthogonal rotation of the previously whitened data through a fixed-point iteration scheme that make the most of a measure of the non-Gaussian distribution of the rotated components.

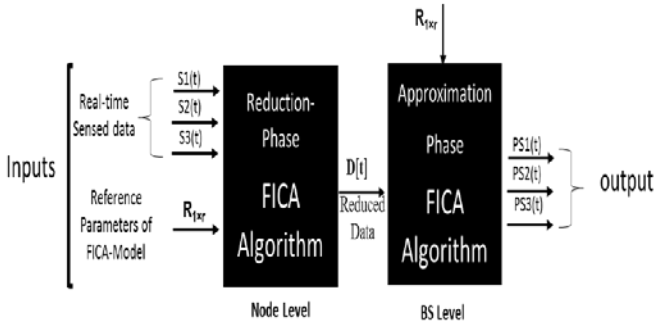

Fig. 5. General Block Diagram of FICA Algorithm.

This section provides a detailed description of the FICA algorithm to reduce the amount of data transferred. Fig. 5 shows the block diagram of WSN data reduction based on the FICA algorithm with a general structure consisting of two phases, namely the reduction phase at the sensor node level and the approximation phase at the BS level. The main objective of training the FICA model is to determine the reference parameters R_(1×r), which are then stored on the sensor node and the same copy is stored on BS. At the node level, the new data S1(t), S2(t), and S3(t) acquired in real-time are reduced by applying the FICA algorithm before transmission and then the reduced data D(t) is sent to BS. After that, the originally acquired data PS1(t), PS2(t), and PS3(t) are estimated by the approximation phase at BS by applying FICA with the same reference parameters R(1×r) used to reduce the node-level data. As mentioned earlier, FICA is used to reduce the packet size of the sensor node. The detailed description of the data reduction-based FICA algorithm is given in the following pseudocode

// FICA

Inputs: Training data $\bar{S}_{mXn}[T]$.//*S1, S2, S3*

Output: $D(t)$

Begin:

Load the training data $\bar{S}_{mXn}[T]$.

Apply FICA ($\bar{S}_{mXn}[T]$) and calculate the eigenvector array $R_{n\times r}$**.** Then**,** decreases the eigenvector array to $\boldsymbol{R_{1\times r}}$ **,**and preserved a copy at the node and transfers one copy to the BS.

Standardizes the current measured data $\bar{S}_{1\times n}[t]$, then decreases it before transferring via the following Equation:

$$D_{1\times r}[t] = \bar{S}_{1\times n}[t] \times R_{1\times r}$$

Send the diminished data $\boldsymbol{D_{1\times r}[t]}$ to the BS.

Approximations data at BS by applying

$$\hat{S}_{1\times n}[t] = D_{1\times r}[t] \times R_{r\times n}$$

End Algorithm

III. Real-Time Datasets

The considered algorithms are evaluated on different benchmark real-time datasets, as described in the following subsections. It's important to note that, usually only part of the data from specific nodes of these datasets are used to assess the performance of current data reduction methods in WSN [17][18][19][20][11][21][22][23]. The reason is that most data reduction methods focus on reducing the amount of transferred data without considering how this data is forwarded to the CH /BS. In other words, they assume that the sensor nodes can directly transmit the sensed data to the CH /BS. The selected algorithms NNF, NNST, EDCD2, LRMV, and FICA are evaluated on real-time datasets as shown below:

A. Data 1-AirQ

Data 1- Air Quality (AirQ) is a WSN data set, including air pressure, humidity and temperature sensors. These sensor data have been collected by 56 sensor nodes in year 2017 at Krakow, Poland. For more information, see the main source [24]. Fig. 6 shows the structure of Data1- AirQ. In addition, some samples of sensors value provided in Tables I to V.

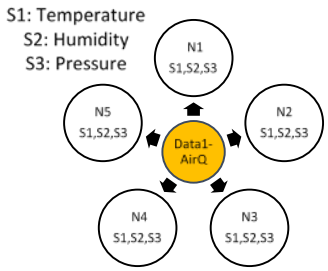

Fig. 6. Structure of Data1- AirQ.

TABLE I. Some Data Samples of Node1 – Data1- AirQ

Sample	Temperature	Humidity	Pressure
1	6	92	101906
2	6	92	101869
3	5	94	101837
4	5	92	101834
5	4	94	101832
6	5	94	101833
7	9	78	101842
8	11	66	101831
9	15	50	101798
10	17	42	101745

TABLE II. Some Data Samples of Node2 – Data1- AirQ

Sample	Temperature	Humidity	Pressure
1	15	90	101514
2	15	90	101516
3	15	90	101530
4	15	92	101555
5	16	90	101577
6	16	90	101595
7	19	91	101601
8	19	88	101592
9	20	82	101571
10	21	81	101541

TABLE III. Some Data Samples of Node3 – Data1- AirQ

Sample	Temperature	Humidity	Pressure
1	14	88	100825
2	14	92	100797
3	14	94	100781
4	14	94	100805
5	14	92	100761
6	14	90	100795
7	15	92	100822
8	15	90	100839
9	16	85	100834
10	18	77	100849

TABLE IV. Some Data Samples of Node4 – Data1- AirQ

Sample	Temperature	Humidity	Pressure
1	8	104	101967
2	7	109	101969
3	6	112	101975
4	6	114	101980
5	6	112	101963
6	8	105	101920
7	8	99	101895
8	10	87	101864
9	11	82	101837
10	12	78	101853

TABLE V. SOME DATA SAMPLES OF NODE5 – DATA1- AIRQ

Sample	Temperature	Humidity	Pressure
1	5	94	102384
2	4	100	102396
3	3	94	102413
4	3	100	102415
5	2	97	102453
6	2	94	102510
7	4	94	102564
8	6	88	102593
9	9	82	102606
10	11	76	102603

B. Data 2-ARHO

Data2- American River Hydrologic Observatory (ARHO) is a WSNs data set, including soil temperature, relative humidity, snow depth sensors, etc. These sensor data have been collected by 130 spatially distributed sensor nodes in the river basin of the United States. Period: the water year 2014 to the water year 2017. For more information, see the main source [25]. Fig. 7 shows the structure of Data 2- ARHO, also some examples of sensor values for all used nodes in Tables VI to X.

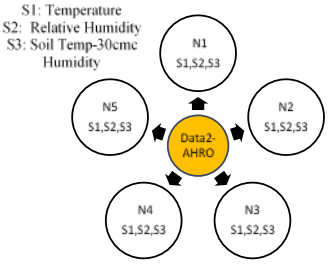

Fig. 7. Structure of Data-2 ARHO.

TABLE VI. SOME DATA SAMPLES OF NODE1 – DATA2- ARHO

Sample	Temperature (℃)	Relative Humidity (%)	Soil Temp-30cmc
1	13.2	38.462	11.5
2	12.52	39.867	11.6
3	12.01	40.756	11.6
4	11.45	42.309	11.7
5	11.04	43.095	11.8
6	10.16	45.276	11.9
7	9.52	46.607	11.9
8	8.98	47.843	12
9	8.31	47.911	12
10	7.87	50.451	12

TABLE VII. SOME DATA SAMPLES OF NODE2 – DATA2- ARHO

Sample	Temperature (℃)	Relative Humidity (%)	Soil Temp-30cmc
1	13.91	37.442	15.3
2	13.65	38.112	15.6
3	13.18	39.249	15.9
4	12.54	41.042	16.1
5	12.2	42.072	16.4
6	11.16	44.384	16.6
7	10.39	46.591	16.8
8	9.38	49.097	16.9
9	8.81	48.684	17.1
10	8.4	50.978	17.2

TABLE VIII. SOME DATA SAMPLES OF NODE3 – DATA2- ARHO

Sample	Temperature (℃)	Relative Humidity (%)	Soil Temp-30cmc
1	13.44	40.587	13.9
2	12.76	42.232	14.1
3	12.07	44.779	14.1
4	11.66	46.026	14.2
5	11.24	47.229	14.4
6	10.72	48.717	14.5
7	10.16	49.661	14.5
8	9.9	50.147	14.6
9	9.43	50.694	14.7
10	8.86	51.858	14.7

TABLE IX. SOME DATA SAMPLES OF NODE4– DATA2- ARHO

Sample	Temperature (℃)	Relative Humidity (%)	Soil Temp-30cmc
1	13.46	38.555	11.1
2	13.12	39.243	11.1
3	12.56	40.556	11.1
4	12.07	41.831	11.2
5	11.66	43.487	11.2
6	11.34	44.469	11.2
7	10.92	45.431	11.3
8	10.37	46.87	11.3
9	9.84	48.325	11.3
10	9.46	47.626	11.3

TABLE X. SOME DATA SAMPLES OF NODE5 – DATA2- ARHO

Sample	Temperature (℃)	Relative Humidity (%)	Soil Temp-30cmc
1	13.12	39.374	11.4
2	12.58	40.493	11.4
3	12.33	41.796	11.5
4	11.98	42.819	11.5
5	11.04	44.845	11.6
6	10.66	45.966	11.6
7	10.3	46.642	11.6
8	9.64	46.653	11.7
9	9.25	48.156	11.7
10	8.95	49.285	11.7

C. Data 3- GSB

Data3- "Grand St. Bernard (GSB)" is WSN data set, which it was gathered by deployed 23 sensors to observer the measurement characteristics of the environmental in the "Grand Saint Bernard Pass" between Switzerland and Italy. The sensors are relative humidity, surface temperature, and ambient temperature [26]. Fig. 8 shows the structure of Data 3- GSB. Some examples of the sensor values of all the nodes used can be found in Tables XI to XV.

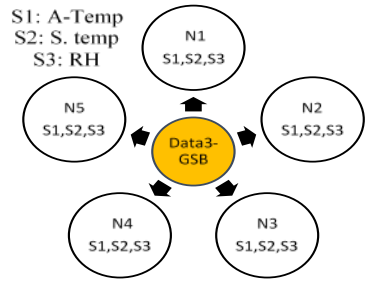

Fig. 8. Structure of Data-3 GSB.

TABLE XI. SOME TEMPERATURE SENSOR DATA SAMPLES OF NODE1_ DATA 3- GSB

Sample	A. Temperature (℃)	Relative Humidity (%)	S. Temperature (℃)
1	12.915	51.7785	12.881
2	12.53	52.3073	12.183
3	12.56	50.2515	12.5995
4	13.1533	51.1487	13.8287
5	12.65	51.121	13.131
6	12.81	51.4583	13.7457
7	12.52	51.2055	12.412
8	12.6267	50.2657	12.475
9	12.52	50.19	12.412
10	12.59	51.2353	12.058

TABLE XII. SOME TEMPERATURE SENSOR DATA SAMPLES OF NODE2– DATA 3- GSB

Sample	A. Temperature (℃)	Relative Humidity (%)	S. Temperature (℃)
1	6.48	84.963	4.225
2	6.36	85.505	4.537
3	6.3	86.08	5.35
4	6.23	86.558	5.975
5	6.18	86.904	6.475
6	6.18	87.128	6.6
7	6.16	87.257	6.725
8	6.16	87.39	6.787
9	6.15	87.499	6.85
10	6.22	87.649	6.787

TABLE XIII. SOME TEMPERATURE SENSOR DATA SAMPLES OF NODE3– DATA 3- GSB

Sample	A. Temperature (℃)	Relative Humidity (%)	S. Temperature (℃)
1	10.92	87.232	11.412
2	10.94	87.877	11.35
3	10.9	87.433	11.412
4	10.9	87.296	11.35
5	10.88	86.946	11.287
6	10.91	87.436	11.225
7	10.93	88.057	11.225
8	10.91	88.211	11.287
9	10.85	87.192	11.162
10	10.88	87.406	11.35

TABLE XIV. SOME TEMPERATURE SENSOR DATA SAMPLES OF NODE4– DATA 3- GSB

Sample	A. Temperature (℃)	Relative Humidity (%)	S. Temperature (℃)
1	0.474802	7.08262	0.274989
2	0.482715	7.076051	0.33434
3	0.473219	7.095914	0.413473
4	0.471637	7.146639	0.47773
5	0.477176	7.171882	0.507405
6	0.474011	7.172753	0.53708
7	0.476385	7.168242	0.561849
8	0.481133	7.193169	0.561849
9	0.474011	7.231628	0.561849
10	0.471637	7.247771	0.561849

TABLE XV. SOME TEMPERATURE SENSOR DATA SAMPLES OF NODE5– DATA3- GSB

Sample	A. Temperature (℃)	Relative Humidity (%)	S. Temperature (℃)
1	4.66891	37.29656	4.879268
2	4.677461	37.57233	4.852759
3	4.660359	37.38249	4.879268
4	4.660359	37.32392	4.852759
5	4.651808	37.17427	4.825823
6	4.664635	37.38378	4.799315
7	4.673186	37.64929	4.799315
8	4.664635	37.71513	4.825823
9	4.638981	37.27945	4.772379
10	4.651808	37.37095	4.852759

D. Data 4- Intel

Data4- "Intel Berkeley Research Lab (IBRL)" is a WSN data-set, which it was gathered by deployed 54 Mica2Dot sensor nodes at "Intel's research Lab", University of Berkeley. The wireless network consisted of. The WSN includes various sensors: voltage, temperature, light, and humidity [27]. Fig. 9 shows the structure of Data 4- Intel, also some examples of sensor values for all the nodes used in Tables VI to XX.

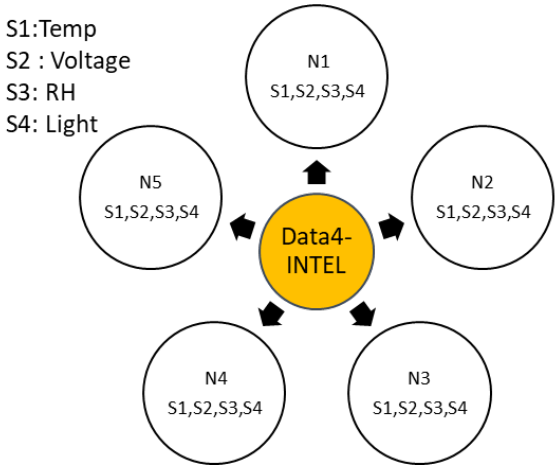

Fig. 9. Structure of Data4-Intel.

TABLE XVI. SOME TEMPERATURE SENSOR DATA SAMPLES OF NODE1 – DATA4-INTEL

Sample	Temperature (℃)	Relative Humidity (%)	Light (Lux)	Voltage (V)
1	19.9884	37.0933	45.08	2.034
2	19.9884	37.0933	45.08	2.6996
3	19.3024	38.4629	45.08	2.6874
4	19.1652	38.8039	45.08	2.6874
5	19.175	38.8379	45.08	2.6996
6	19.1456	38.9401	45.08	2.6874
7	19.1652	38.872	45.08	2.6874
8	19.1652	38.8039	45.08	2.6874
9	19.1456	38.8379	45.08	2.6996
10	19.1456	38.872	45.08	2.6874

TABLE XVII. SOME TEMPERATURE SENSOR DATA SAMPLES OF NODE2 – DATA4-INTEL

Sample	Temperature (°C)	Relative Humidity (%)	Light (Lux)	Voltage (V)
1	89.5488	29.2581	11.96	1.9537
2	19.567	39.6878	121.44	2.6753
3	19.5376	39.7557	121.44	2.6753
4	19.4788	39.6878	121.44	2.6633
5	19.4494	39.7217	121.44	2.6753
6	19.4984	39.586	121.44	2.6753
7	19.4788	39.5521	121.44	2.6633
8	19.4592	39.5181	121.44	2.6753
9	19.4494	39.5521	121.44	2.6753
10	19.4788	39.4162	121.44	2.6874

TABLE XVIII. SOME TEMPERATURE SENSOR DATA SAMPLES OF NODE3 – DATA4-INTEL

Sample	Temperature (°C)	Relative Humidity (%)	Light (Lux)	Voltage (V)
1	22.2816	43.8515	41.4	2.5935
2	22.2718	43.9844	41.4	2.5935
3	22.2718	43.8848	41.4	2.5935
4	22.5168	48.1243	382.72	2.32
5	22.2816	43.918	41.4	2.5935
6	22.2718	43.7186	41.4	2.5935
7	22.262	43.519	41.4	2.6049
8	22.2718	43.519	41.4	2.5935
9	22.2718	43.7186	41.4	2.5935
10	22.6148	47.7013	264.96	2.32

TABLE XIX. SOME TEMPERATURE SENSOR DATA SAMPLES OF NODE4 – DATA4-INTEL

Sample	Temperature (°C)	Relative Humidity (%)	Light (Lux)	Voltage (V)
1	19.1848	38.9742	108.56	2.6874
2	19.0084	39.4502	108.56	2.6874
3	18.9398	39.6539	108.56	2.6996
4	18.8712	40.0607	108.56	2.6874
5	18.8516	40.0945	108.56	2.6996
6	18.8418	40.2976	108.56	2.6996
7	18.8418	40.2976	108.56	2.6996
8	18.832	40.2976	108.56	2.6996
9	18.8222	40.2638	108.56	2.6996
10	18.8124	40.2976	108.56	2.6874

TABLE XX. SOME TEMPERATURE SENSOR DATA SAMPLES OF NODE5 – DATA4-INTEL

Sample	Temperature (°C)	Relative Humidity (%)	Light (Lux)	Voltage (V)
1	19.3612	39.8235	75.44	2.67532
2	19.273	39.9252	75.44	2.67532
3	18.9888	40.9392	75.44	2.67532
4	18.9398	40.8718	75.44	2.67532
5	18.9006	40.973	75.44	2.68742
6	18.9496	40.9055	75.44	2.67532
7	18.9594	41.3098	75.44	2.67532
8	18.9496	41.3771	75.44	2.67532
9	18.9496	41.0404	75.44	2.67532
10	18.93	40.9055	75.44	2.67532

IV. PERFORMANCE METRICS

A. Accuracy

Accuracy is the overall average of absolute error for all selected nodes from the same dataset as defined below:

$$Accuracy = \frac{\sum_{k=1}^{L} AEPN(k)}{L} \quad (1)$$

$$\text{AEPN(k)} = \frac{\sum_{i=1}^{N} |AEPS(i)|}{N}, \quad (2)$$

$$AEPS(i) = \frac{\sum_{j=1}^{M} |SV(j) - RV(j)|}{M} \quad (3)$$

Where *K= {1, 2,..L}*, *L* is the number of nodes., *AEPN* is the average of absolute error for all samples transmitted by the node (*k*) , *SV* is the sensor value at the sensor node, *RV* is the received value at *BS*, and *AEPS*(i) is the mean Absolute error for sensor (i), i = *{1, 2...N}*, *N* is the number of sensors, *M* is the number of samples transmitted by the node (*k*).

B. Data Reduction Ratio %

$$DR\% = \left(1 - \frac{Sd_{redc}}{Sd_{Lenght}}\right) \times 100 \quad (4)$$

where DR is the ratio of the reduced data, Sd_{redc} is the size of transferred data after reduction and Sd_{Lenght} is the size of the unreduced transmission samples.

C. Total Energy *Consumption*

$$TE_{Directly} = D_S \times N_{of\ S} \times C_E PByte \quad (5)$$

$$TE_{DR} = RD_S \times N_{of\ S} \times C_E P\ Byte \quad (6)$$

Where: $TE_{Directly}$ is the Total Energy consumed in case of the Direct transmission, D_s is the mean Data size, $N_{of\ S}$ is the mean Number of Samples, $C_E\ P\ Byte$ is the mean Cost Energy Per Byte, RD_S is the mean Data Reduction.

V. SIMULATION AND RESULTS

Fig. 10 shows the accuracy of the applied algorithms EDCD2, FICA, NNF, NNTS, and LRMV for all selected nodes N1, N2, N5 from the DATA1-AIRQ dataset. From the results, the EDCD2 algorithm has the best accuracy compared to the other algorithms FICA, NNF, NNTS, and LRMV. The reason for this is the average total absolute error which has the lowest value of 5.48 when EDCD2 is used for all nodes. Moreover, the algorithms FICA and LRMV have the worst performance in terms of accuracy, and the average absolute errors are 62.30 and 20.13, respectively. Table A1, Table A2, Table A3, Table A5 (see Appendix) showed the average of absolute error for all samples transmitted by the nodes (N1-N5) of the applied algorithms EDCD2, FICA, NNF, NNTS, and LRMV, respectively.

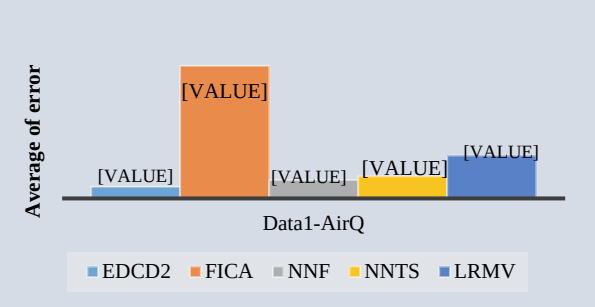

Fig. 10. Accuracy of Applying Various Algorithms for all Selected Nodes from DATA1-AIRQ.

Fig. 11 shows the accuracy of the applied algorithms EDCD2, FICA, NNF, NNTS, and LRMV for all selected nodes N1, N2, N5 from the DATA2-ARHO dataset. From the results, the EDCD2 algorithm has the best accuracy compared to the other algorithms FICA, NNF, NNTS, and LRMV. The reason for this is the average total absolute error which has the lowest value of 0.199 when EDCD2 is used for all nodes. Moreover, NNTS and NNF algorithms have the worst performance in terms of accuracy, and the average absolute errors are 5.38 and 5.62, respectively. In summary, EDCD2 is a threshold-based data reduction algorithm. EDCD2 transmits measurement data only when the relative difference between the current measurement data and the last transmitted data is larger than the threshold value.

Fig. 12 shows the accuracy of the applied EDCD2, FICA, NNF, NNTS, and LRMV algorithms for all selected nodes N1, N2, N5 from the DATA3-GSB dataset. From the results, the EDCD2 algorithm has been shown to have the best accuracy compared with the other algorithms, FICA, NNF, NNTS, and LRMV. The reason is related to the overall average absolute error, which is the lowest value of 0.30 for applied EDCD2 for all nodes. Furthermore, the FICA and NNF algorithms have the worst performance in terms of accuracy, and the average absolute errors are 3.84 and 1.34, respectively.

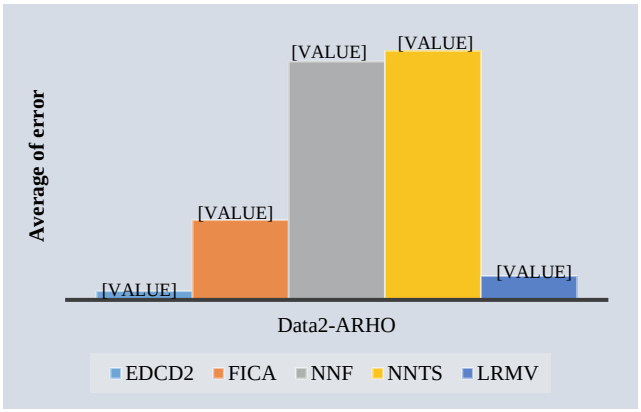

Fig. 11. Accuracy of Applying Various Algorithms for all Selected Nodes from DATA2-ARHO.

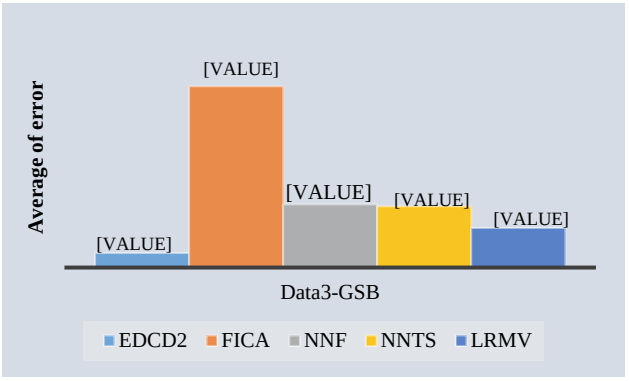

Fig. 12. Accuracy of Applying Various Algorithms for all Selected Nodes from DATA3-GSB.

Fig. 13 shows the accuracy of the applied algorithms EDCD2, FICA, NNF, NNTS, and LRMV for all selected nodes N1, N2, N5 from the DATA4- INTEL dataset. From the results, the NNF algorithm has the best accuracy compared to the other algorithms EDCD2, FICA, NNTS and LRMV. The reason for this is the average total absolute error which has the lowest value of 1.01 when NNF is used for all nodes. Moreover, the algorithms FICA and LRMV have the worst performance in terms of accuracy, and the average absolute errors are 28.68 and 1.54, respectively.

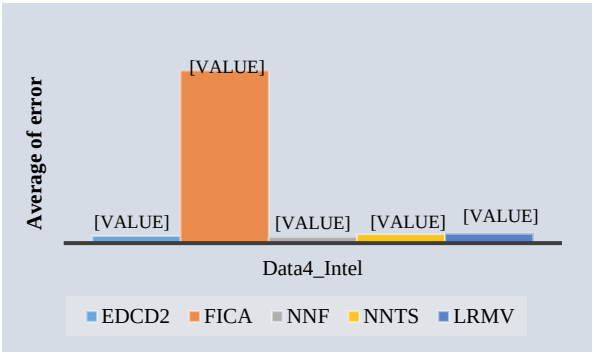

Fig. 13. Accuracy of Applying Various Algorithms for all Selected Nodes from DATA4-INTEL.

Fig. 14 shows the average of data reduction ratio percentage for applying various algorithms for different datasets. The studied algorithms are EDCD2, FICA, NNF, NNTS, and LRMV. The selected datasets are Data1-AirQ, Data2-ARHO, Data3-GSB, and Data4_Intel. From these results, the average data reduction percentage for applied EDCD2, FICA, NNF, NNTS, and LRMV algorithms through a real-time dataset named Data1-AirQ is 33%, 33%, 67%, 67%, and 33%, respectively. It is noted that the NNF and NNTS algorithms have the highest data reduction. By referring to Fig. 10 both algorithms, NNF and NNTS, have acceptable accuracy and the lowest error has been shown by applying EDCD2. In the same way, the average data reduction percentage for applied NNF, NNTS, EDCD2, LRMV, and FICA algorithms through a real-time dataset named Data2-ARHO is 67%, 67%, 56%, 33%, and 67%, respectively. Although NNF and NNTS algorithms achieved the highest data reduction ratio, both NNF and NNTS have the highest error and worst performance in terms of accuracy as shown in Fig. 11 The average data

reduction percentage for applied NNF, NNTS, EDCD2, LRMV, and FICA algorithms through a real-time dataset named Data3-GSB is 67%, 67%, 67%, 33%, and 33%, respectively. It is noted that the NNF, NNTS and EDCD2 algorithms have the highest data reduction, by referring to Fig. 12 the lowest error has been shown by applying EDCD2. FICA showed the worst performance in terms of accuracy, with the highest errors. The average data reduction percentage for applied NNF, NNTS, EDCD2, LRMV, and FICA algorithms through a real-time dataset named Data4_Intel is 50%, 50%, 83%, 25%, and 75%, respectively. It is worth noting that the EDCD2 algorithm achieves the highest data reduction. By referring to Fig. 14, Tables XXI, XXII both NNF, NNTS, EDCD2, and LRMV algorithms have acceptable accuracy, and the highest error has been shown by applying FICA.

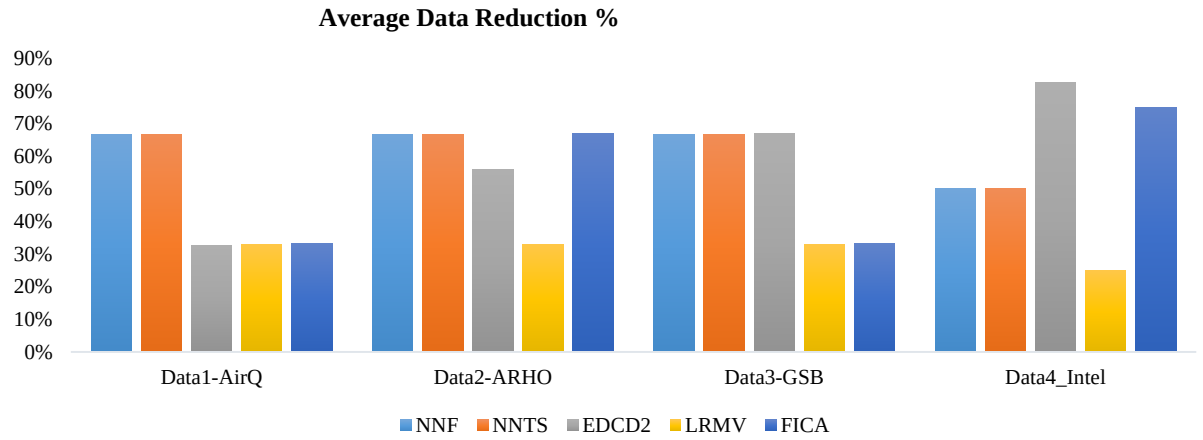

Fig. 14. Average of Data Reduction Ratio % for Applying Various Algorithms for Different Datasets.

TABLE XXI. TOTAL ENERGY CONSUMPTION BY APPLYING THE SELECTED ALGORITHMS VS WITHOUT ALGORITHMS

	NNF	NNTS	EDCD2	LRMV	FICA	Without algorithm
Data1-AirQ	165900	165900	335644.2	333459	331800	497700
Data2-ARHO	165900	165900	219718	333459	164241	497700
Data3-GSB	711000	711000	706592.2	1429110	1422000	2133000
Data4_Intel	1422000	1422000	491443.2	2133000	711000	2844000

TABLE XXII. THE PERCENTAGE OF SAVED ENERGY BY APPLYING THE SELECTED ALGORITHMS

	NNF	NNTS	EDCD2	LRMV	FICA
Data1-AirQ	67%	67%	33%	33%	33%
Data2-ARHO	67%	67%	56%	33%	67%
Data3-GSB	67%	67%	67%	33%	33%
Data4_Intel	50%	50%	83%	25%	75%

VI. CONCLUSION

The impact of data reduction methods on WSN performance is investigated in this paper, using a set of real-time datasets. Simulation tests are performed in MATLAB for different methods to reduce the amount of data sent. The selected algorithms NNF, NNST, EDCD2, LRMV, and FICA are evaluated using real-time data sets. The performance metrics measured are energy consumption, data accuracy, and percentage of data reduction. The results of the study show that the selected algorithm helps to reduce the amount of transmitted data and energy consumption, and each algorithm performs differently depending on the dataset used.

ACKNOWLEDGMENT

"The writers would like to thank University Polyethnic of Bucharest (UPB) for their support to carry out this study."

REFERENCES

[1] M. K. Hussein, "Data Reduction Algorithms for Wireless Sensor Networks Applications: Review," in 2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), 2021, pp. 1–7.

[2] M. I. Husni, M. K. Hussein, N. A. M. Alduais, J. Abdullah, and I. Marghescu, "Performance of Various Algorithms to Reduce the Number of Transmitted Packets by Sensor Nodes in Wireless Sensor Network," in 2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), 2019, pp. 1–7.

[3] N. Alduais et al., "An Efficient IoT-based Smart Water Meter System of Smart City Environment," Artic. Int. J. Adv. Comput. Sci. Appl., vol. 12, no. 8, p. 2021, 2021.

[4] N. A. M. Alduais, J. Abdullah, and A. Jamil, "APRS: adaptive real-time payload data reduction scheme for IoT/WSN sensor board with multivariate sensors," Int. J. Sens. Networks, vol. 28, no. 4, pp. 211–229, 2018.

[5] A. Jarwan, A. Sabbah, and M. Ibnkahla, "Data Transmission Reduction Schemes in WSNs for Efficient IoT Systems," IEEE J. Sel. Areas Commun., vol. 37, no. 6, pp. 1307–1324, 2019.

[6] S. Diwakaran, B. Perumal, and K. Vimala Devi, "A cluster prediction model-based data collection for energy efficient wireless sensor network," J. Supercomput., vol. 75, no. 6, pp. 3302–3316, 2019.

[7] E. P. K. Gilbert, B. Kaliaperumal, E. B. Rajsingh, and M. Lydia, “Trust based data prediction, aggregation and reconstruction using compressed sensing for clustered wireless sensor networks,” Comput. Electr. Eng., vol. 72, pp. 894–909, 2018.

[8] Y. Fathy, P. Barnaghi, and R. Tafazolli, “An adaptive method for data reduction in the Internet of Things,” IEEE World Forum Internet Things, WF-IoT 2018 - Proc., vol. 2018-Janua, pp. 729–735, 2018.

[9] J. Abdullah, M. K. Hussien, N. A. M. Alduais, M. I. Husni, and A. Jamil, “Data Reduction Algorithms based on Computational Intelligence for Wireless Sensor Networks Applications,” in 2019 IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE), 2019, pp. 166–171.

[10] N. A. M. Alduais, J. Abdullah, A. Jamil, and H. Heidari, “Performance Evaluation of Real-Time Multivariate Data Reduction Models for Adaptive-Threshold in Wireless Sensor Networks,” IEEE Sensors Lett., vol. 1, no. 6, pp. 1–4, 2017.

[11] N. A. M. ALDUAIS and A. A. J. JIWA ABDULLAH, “RDCM : An Efficient Real-Time Data Collection Model for IoT / WSN Edge With Multivariate Sensors,” IEEE Access, vol. 7, pp. 89063–89082, 2019.

[12] Fit Data with a Shallow Neural Network - MATLAB & Simulink - MathWorks United Kingdom, “No Title,” Mathworks.com, 2017. [Online]. Available: https://www.mathworks.com/help/nnet/gs/fit-data-with-a-neura, 2017. .

[13] “Linear Regression with Multiple Variables | Machine Learning, Deep Learning, and Computer Vision.” [Online]. Available: https://www.ritchieng.com/multi-variable-linear-regression/. [Accessed: 25-Sep-2021].

[14] N. A. M. Alduais, “An Efficient Data Collection Algorithms for IoT Sensor Board,” 2016.

[15] M. I. Al-Qinna and S. M. Jaber, “Predicting soil bulk density using advanced pedotransfer functions in an arid environment,” Trans. ASABE, vol. 56, no. 3, pp. 963–976, 2013.

[16] A. Hyvärinen and E. Oja, “A Fast Fixed-Point Algorithm for Independent Component Analysis,” Neural Comput., vol. 9, no. 7, pp. 1483–1492, 1997.

[17] AISSMS Institute of Information Technology and Institute of Electrical and Electronics Engineers, “2020 International Conference on Emerging Smart Computing and Informatics (ESCI) : AISSMS Institute of Information Technology, Pune, India. Mar 12-14, 2020.,” pp. 103–108, 2020.

[18] L. Mesin, S. Aram, and E. Pasero, “A Neural Data-Driven Approach to increase Wireless Sensor Networl(s’ lifetime,” pp. 1–3, 2014.

[19] M. A. Rassam, A. Zainal, and M. A. Maarof, “An adaptive and efficient dimension reduction model for multivariate wireless sensor networks applications,” Appl. Soft Comput. J., vol. 13, no. 4, pp. 1978–1996, 2013.

[20] H. Harb et al., “Industrial Process Monitoring To cite this version :,” 2019.

[21] L. Tan and M. Wu, “Data Reduction in Wireless Sensor Networks: A Hierarchical LMS Prediction Approach,” IEEE Sens. J., vol. 16, no. 6, pp. 1708–1715, 2016.

[22] C. Carvalho, D. G. Gomes, N. Agoulmine, and J. N. de Souza, “Improving prediction accuracy for WSN data reduction by applying multivariate spatio-temporal correlation,” Sensors, vol. 11, no. 11, pp. 10010–10037, 2011.

[23] Y. Deng, C. Han, J. Guo, and L. Sun, “Temporal and spatial nearest neighbor values based missing data imputation in wireless sensor networks,” Sensors, vol. 21, no. 5, pp. 1–24, 2021.

[24] UCI Machine Learning Repository, “No Title,” Air Quality Data Set", Archive.ics.uci.edu, 2018. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Air+quality. [Accessed: 23- Jan- 2018]., 2018. .

[25] “Dryad Data -- Snow depth, air temperature, humidity, soil moisture and temperature, and solar radiation data from the basin-scale wireless-sensor network in American River Hydrologic Observatory (ARHO).” [Online]. Available: https://datadryad.org/stash/dataset/doi:10.6071/M39Q2V. [Accessed: 24-Sep-2021].

[26] “Index of /~rossi/papers/CS_2012.” [Online]. Available: http://www.dei.unipd.it/~rossi/papers/CS_2012/. [Accessed: 24-Sep-2021].

[27] Intel Lab data, “Intel-dataSet,” [Online]. Available: http://db.csail.mit.edu/labdata/labdata.html. [Accessed: 23- Jan- 2018]., 2018.

APPENDIX

TABLE A1. AVERAGE OF ABSOLUTE ERROR RESULTS OF APPLYING EDCD2 ALGORITHM FOR ALL NODES WITH DIFFERENT DATASETS

EDCD2																	
Nodes	Data1-AirQ				Data2-ARHO				Data3-GSB				Data4_Intel				
	Temperature	Humidity	Pressure	Mean	Temperature (°C)	soilT_30cmC	Relative Humidity (%)	Mean	A. Temperature (°C) S.	Temperature (°C)	Relative Humidity (%)	Mean	Temperature (°C)	Voltage(v)	Relative Humidity (%)	Light (Lux)	Mean
N1	0.011	0.469	20.657	7.046	0.126	0.036	0.425	0.196	0.093	0.112	0.565	0.257	0.304	0.005	0.409	1.834	0.749
N2	0.011	0.469	20.657	7.046	0.111	0.062	0.397	0.190	0.092	0.113	0.823	0.342	0.109	0.003	0.152	1.797	0.651
N3	0.059	0.313	8.548	2.974	0.110	0.038	0.411	0.186	0.100	0.107	0.780	0.329	0.389	0.002	0.149	0.879	0.343
N4	0.068	0.326	10.167	3.520	0.130	0.024	0.489	0.214	0.096	0.100	0.719	0.305	0.218	0.004	0.291	7.367	2.554
N5	0.014	0.535	20.044	6.864	0.119	0.032	0.472	0.208	0.097	0.102	0.704	0.301	0.194	0.006	0.320	4.542	1.623
Average	**0.033**	**0.423**	**16.015**	**5.490**	**0.119**	**0.038**	**0.439**	**0.199**	**0.096**	**0.107**	**0.718**	**0.307**	**0.243**	**0.004**	**0.264**	**3.284**	**1.184**

TABLE A2. AVERAGE OF ABSOLUTE ERROR RESULTS OF APPLYING FICA ALGORITHM FOR ALL NODES WITH DIFFERENT DATASETS

FICA																	
Nodes	Data1-AirQ				Data2-ARHO				Data3-GSB				Data4_Intel				
	Temperature	Humidity	Pressure	Mean	Temperature (°C)	soilT_30cmc	Relative Humidity (%)	Mean	A. Temperature (°C)	S. Temperature (°C)	Relative Humidity (%)	Mean	Temperature (°C)	Voltage(v)	Relative Humidity (%)	Light (Lux)	Mean
N1	3.193	3.631	152.485	53.103	2.206	1.198	2.152	1.852	0.883	0.490	6.779	2.717	1.771	0.017	2.209	128.169	43.465
N2	3.193	3.631	152.485	53.103	2.252	1.897	3.297	2.482	1.458	1.644	10.950	4.684	1.004	0.029	2.623	109.780	37.477
N3	4.410	8.748	143.465	52.208	2.394	1.444	1.316	1.718	1.088	0.509	9.897	3.831	17.764	0.130	4.089	115.559	39.926
N4	3.004	3.234	170.109	58.782	2.063	0.827	1.250	1.380	1.102	0.428	9.115	3.548	1.290	0.040	1.901	36.201	12.714
N5	1.565	6.314	275.148	94.343	1.987	1.178	1.583	1.583	1.568	1.178	10.622	4.456	1.173	0.015	1.915	27.623	9.851
Average	3.073	5.112	178.738	62.308	2.180	1.309	1.920	1.803	1.220	0.850	9.473	3.847	4.601	0.046	2.547	83.467	28.687

TABLE A3. AVERAGE OF ABSOLUTE ERROR RESULTS OF APPLYING NNF ALGORITHM FOR ALL NODES WITH DIFFERENT DATASETS

NNF																	
Nodes	Data1-AirQ				Data2-ARHO				Data3-GSB				Data4_Intel				
	Temperature	Humidity	Pressure	Mean	Temperature (°C)	soilT_30cmc	Relative Humidity (%)	Mean	A. Temperature (°C)	S. Temperature (°C)	Relative Humidity (%)	Mean	Temperature (°C)	Voltage(v)	Relative Humidity (%)	Light (Lux)	Mean
N1	0.000	16.472	4.318	6.930	0.000	14.270	1.768	5.346	0.000	1.518	2.860	1.459	0.000	2.190	0.011	0.000	0.734
N2	0.000	20.839	5.614	8.818	0.000	13.882	2.839	5.574	0.000	1.882	3.242	1.708	0.000	7.855	0.035	0.000	2.630
N3	0.000	25.032	5.963	10.332	0.000	14.229	2.449	5.559	0.000	1.470	2.359	1.276	0.000	2.256	0.045	0.000	0.767
N4	0.000	23.722	5.138	9.620	0.000	13.814	1.261	5.025	0.000	1.340	1.800	1.047	0.000	1.323	0.034	0.000	0.452
N5	0.000	18.361	3.503	7.288	0.000	14.688	1.567	5.418	0.000	1.456	2.184	1.213	0.000	1.381	0.009	0.000	0.463
Average	0.000	20.885	4.907	8.597	0.000	14.177	1.977	5.384	0.000	1.533	2.489	1.341	0.000	3.001	0.027	0.000	1.009

TABLE A4. AVERAGE OF ABSOLUTE ERROR RESULTS OF APPLYING NNTS ALGORITHM FOR ALL NODES WITH DIFFERENT DATASETS

NNTS																	
Nodes	Data1-AirQ				Data2-ARHO				Data3-GSB				Data4_Intel				
	Temperature	Humidity	Pressure	Mean	Temperature (°C)	soilT_30cmc	Relative Humidity (%)	Mean	A. Temperature (°C)	S. Temperature (°C)	Relative Humidity (%)	Mean	Temperature (°C)	Voltage(v)	Relative Humidity (%)	Light (Lux)	Mean
N1	0.000	14.871	4.838	6.570	0.000	15.571	1.860	5.810	0.000	2.840	1.682	1.507	0.000	2.295	0.013	0.000	0.769
N2	0.000	36.685	17.200	17.962	0.000	14.526	2.835	5.787	0.000	2.464	1.476	1.313	0.000	9.455	0.280	0.000	3.245
N3	0.000	25.774	10.295	12.023	0.000	14.771	2.516	5.762	0.000	2.222	1.533	1.252	0.000	4.609	0.370	0.000	1.660
N4	0.000	19.630	5.119	8.250	0.000	14.295	1.200	5.165	0.000	1.837	1.516	1.118	0.000	3.719	0.038	0.000	1.252
N5	0.000	18.334	3.423	7.252	0.000	15.152	1.640	5.597	0.000	2.272	1.617	1.296	0.000	1.385	0.009	0.000	0.465
Average	0.000	23.059	8.175	10.411	0.000	14.863	2.010	5.624	0.000	2.327	1.565	1.297	0.000	4.293	0.142	0.000	1.478

TABLE A5. AVERAGE OF ABSOLUTE ERROR RESULTS OF APPLYING LRMV ALGORITHM FOR ALL NODES WITH DIFFERENT DATASETS

LRMV																	
Nodes	Data1-AirQ				Data2-ARHO				Data3-GSB				Data4_Intel				
	Temperature	Humidity	Pressure	Mean	Temperature (°C)	soilT_30cmc	Relative Humidity (%)	Mean	A. Temperature (°C)	S. Temperature (°C)	Relative Humidity (%)	Mean	Temperature (°C)	Voltage(v)	Relative Humidity (%)	Light (Lux)	Mean
N1	0.000	69.680	0.000	23.227	0.000	1.212	0.000	0.404	0.000	5.211	0.000	1.737	0.000	1.771	0.017	0.000	0.596
N2	0.000	69.680	0.000	23.227	0.000	2.132	0.000	0.711	0.000	1.933	0.000	0.644	0.000	1.004	0.029	0.000	0.345
N3	0.000	133.045	0.000	44.348	0.000	2.615	0.000	0.872	0.000	1.460	0.000	0.487	0.000	17.764	0.130	0.000	5.965
N4	0.000	25.408	0.000	8.469	0.000	0.845	0.000	0.282	0.000	1.411	0.000	0.470	0.000	1.290	0.040	0.000	0.444
N5	0.000	4.274	0.000	1.425	0.000	1.278	0.000	0.426	0.000	2.587	0.000	0.862	0.000	1.173	0.015	0.000	0.396
Average	0.000	60.417	0.000	20.139	0.000	1.616	0.000	0.539	0.000	2.520	0.000	0.840	0.000	4.601	0.046	0.000	1.549