iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** V **Month of publication:** May 2022

DOI: https://doi.org/10.22214/ijraset.2022.41838

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue V May 2022- Available at www.ijraset.com

An Improved Stacked Auto Encoder based Data Compression Technique for WSNs

Lithin Kumble[1], Kiran Kumari Patil[2]
[1, 2] *School of Computing and IT, REVA University*

Abstract: ***Because of the limited energy available to sensor nodes in wireless sensor networks (WSNs), data compression is critical in these networks. The majority of the time, data communication results in energy consumption; however, by minimizing data transmission and reception, the lifetime of sensor nodes may usually be extended significantly. To compress sensor data, we present a new Improved Stacked RBM Auto-Encoder model, which is built of two layers: an encode layer and a decode layer, which is described in detail in this work. Data from sensors is compressed and decompressed in the encode layer; data from sensors is reconstructed and compressed in the decode layer. The encode layer and the decode layer are both made up of four conventional Restricted Boltzmann Machines that are used throughout the system (RBMs). We also present an energy optimization strategy that, by trimming the parameters of the model, can further minimize the energy consumption of the model storage and calculation. We evaluate the model's performance by comparing it to the data acquired by Intel Lab in the environment. Assuming that the model's compression ratio is 10, the average Percentage RMS Difference value is 9.84 percent, and the average temperature reconstruction error value is 0.312 degrees Celsius. It is possible to minimize the energy consumption of node communication in WSNs by 92 percent. When compared to the traditional method, the proposed model achieves higher compression efficiency and reconstruction accuracy while maintaining the same compression ratio as the old method. The results of our experiments demonstrate that the new neural network model can not only be applied to data compression for WSNs, but it also has high compression efficiency and an excellent transfer learning capability.***
Keywords: Data Compression; Stacked-Autocoder; transfer learning; energy, consumption optimization

I. INTRODUCTION

Sensor networks ideal for collecting environmental information such as temperature, light, and humidity in difficult-to-reach regions have been discovered through research. Because of the advancement of information technology in recent years, wireless sensor networks (WSNs) have played an increasingly essential role in a wide range of fields and disciplines [1. Unmanned vehicles require a GPS and an accelerometer in order to find themselves, as well as a camera and lidar in order to gather information about their surroundings [2]. In addition, they can forecast the travelling trajectory of objects in the vicinity by combining this multi-modal sensor data. Similarly, environmental monitoring apps require information such as temperature, humidity, wind direction, and so on to function properly. The proliferation of WSN applications and sensor nodes has resulted in a massive rise in the amount of sensing data available. Direct transfer of the sensing data acquired by WSNs to a gateway will not only cost a significant amount of power, but it will also increase the likelihood of transmission mistakes. Because the storage and portable energy resources available to a sensor node are limited, it is critical to develop a data compression strategy for WSNs that is both energy efficient and redundant. These considerations have prompted the development of energy-efficient technologies that reduce energy usage while also extending the lifetime of WSNs. The data gathering methods used by WSNs include direct transmission to the base station, multi-hop forwarding, data aggregation, and modelling with coding, among other things. Data aggregation is a critical technology for data processing in wireless sensor networks. The amount of data that must be transferred can be significantly decreased [3] by aggregating the data that has been collected or received. Existing research on data aggregation algorithms often involves a node comparing the relationship between its perceptual data and the data from the surrounding nodes in order to determine the best technique. The information will not be submitted if it is determined that the data is close in order to reduce the transmission of redundant information. Even while these aggregation algorithms minimize the amount of data transmitted and reduce the energy consumed by the node, they result in the loss of node information. Furthermore, when aggregating, the outlier detection is time-consuming and might result in delays [4, 5, 6, 7, 8]. It is possible to accomplish non-collaborative data compression at the sources using the Slepian-Wolf coding technique, which makes use of dispersed source coding technology. Although theoretically feasible [5, it is not feasible due to a lack of prior understanding of the data correlation structure.

©IJRASET: All Rights are Reserved

Model-based compression algorithms such as APCA [6] and PWLH [7] have also demonstrated high compression ratios in the past. However, because they approximate data with temporal and spatial locality, they cause a loss of sensing data to be recorded. There has been some study that indicates that modelling in conjunction with the encoding approach is preferable for data transmission in WSNs [8]. Initially, it models the sensor node's original sensing data, after which the modeled weights are transferred to the base station, which then utilizes the weights to compress the original sensing data. Compressed sensing (CS) [9] is a new data compression technique for wireless sensor networks. [10] The core concept is that, provided that a signal is sparse or compressible at a particular level, it can be recreated from a limited number of linear measurements that are less than the Shannon–Nyquist limit. Numerous notable advancements in the field of signal processing have resulted from recent research into computational statistics, including unique sub-Nyquist sampling algorithms and a remarkable burst of work in sparse approximation and representation [11,12]. A approach for reducing the sensing data traffic for wireless sensor networks is proposed in Ref. [13], and it does not necessitate any adaptations to the data correlation structure. In Ref. [14], the CS principle was applied as a compression and forwarding technique in order to reduce the amount of data transmitted. When the signal amplitude is high, the current implementation of the CS necessitates the use of a substantial amount of memory space to hold the random sampling operator.

The traditional CS is therefore not directly applicable to large-scale applications, and it will necessarily increase the computational complexity of encoding, so that the compression achieved by the classic compression is restricted by the mobile embedded processor. At the same time, because of the energy constraints, it is difficult to deploy large-scale WSNs. The analogue CS approach [15] is a novel mechanism for sampling and processing sparse signals at a rate that is less than the Nyquist rate. [16] addressed the issue of energy usage for sensor nodes performing CS and DCS when both digital and analogue CS were taken into consideration. To reduce the total amount of energy consumed by a WSN while gathering sensor data from the entire network [17], the least energy CS-based data aggregation problem was investigated. In Ref. [18], the CS-based signal and data collection for WSNs was proposed, as well as a cluster-sparse reconstruction technique for in-network compression in order to achieve precise signal recovery and energy efficiency while maintaining network performance. Although the segmented linear compression method, which employs polynomial approximation in the form of segmentation to reduce the dimension of a high compression ratio, has poor smoothness, poor precision, and abnormal change, it is a viable option for reducing the size of a high compression ratio. The data compression strategies for WSNs that are based on spatiotemporal correlation were summarized in Refs. [19,20]. A method for compressing sensing data for WSNs was proposed in Ref. [21], and it is described as follows: run length coding method. When applied to the same distortion rate as wavelet compression, a technique based on spatial correlation has been proposed in ICACT [22], which saves more energy than wavelet compression in terms of energy savings. However, as previously stated, the shallow learning algorithm's learning ability is extremely limited when it comes to advanced features, as evidenced by the example above.

In addition, because the deep learning method can extract detection information from multiple levels of features, the system has a strong data fitting ability because it is trained to learn deep features from the data. Presently, there are just a few studies on how to compress sensing data for WSNs using a deep learning model, and these studies are limited in scope. In Ref. [23], a strategy for reducing the dimensionality of data was described, which involved the use of principle components analysis (PCA). Reference [24] discusses the integration of machine learning methodologies with CS, in which feed-forward deep neural network topologies are utilized to aid in the reconstruction of CS signals. In Ref. [25], a stacked Auto-Encoder (SAE)-based data compression technique was proposed, which was implemented. Combining the SAE with the cluster routing protocol is what this method is called. If the SAE compression algorithm is used instead of the usual compression technique, the accuracy of data fusion can be improved by 7.5 percent. In Ref. [26], it is recommended to use a deep convolution network for ECG signal reduction, which requires a significant amount of computation. The Restricted Boltzmann Machine (RBM) is a probabilistic model for a density over observed variables that make use of a collection of hidden variables to achieve the desired result (representing presence of features). RBM is a method in which all observed variables are connected to all hidden variables through the use of various parameters. It was frequently employed in the classification and creation of information. The Convolutional RBM (C-RBM) was created in Ref. [27] in order to achieve object detection. It is a version of the RBM model in which weights are shared in order to preserve the spatial structure of images, as opposed to the RBM model.

Previously, in Ref. [28], an algorithm for pre-training Deep Boltzmann Machines (DBMs) was described in detail. CS reconstruction algorithm was proposed in Ref. [29], which consists of two nested inference problems, one on the CS observation-matching problem and the other on the RBM model, and is described in detail in Ref. [30]. When compared to variational autoencoders (VAE) and generative adversarial networks (GAN), RBM has the simplest network structure and requires the fewest number of parameters to operate on. As a result, RBM consumes little computational energy, making it more suitable for use in wireless sensor networks (WSNs) with limited energy.

According to our survey results, there is currently no research on how to directly use RBM to compress the sensing data for WSNs at this time. Currently, we are primarily interested in learning how to compress sensing data using RBM. In our next research, we will also look into how to compress sensing data using VAE, GAN, and other deep learning models. Using the RBM generation model in conjunction with the nonlinear learning method of deep learning theory, we developed the Stacked CAE-AE model, which can compress and reconstruct sensing data from wireless sensor networks. The encoder and decoder of the Stacked CAE-AE model are formed by stacking four ordinary RBMs of different sizes together. Utilizing the mathematical characteristics of the Stacked CAE-AE model, the sensing data is compressed using the model. Within the context of our tests, we investigate and compare the performance of the model against many different compression methods, and we propose an energy optimization strategy for the model to help reduce the amount of energy consumed during the calculation. The following are some of the key contributions made by this paper:

The Stacked CAE is a hybrid model that combines four standard RBMs with the auto-encoder function to compress and reconstruct the sensing data.

1) It is the first hybrid model to be produced. We suggested a new approach of data compression that, when compared to the old methodology,

2) Provides improved reconstruction accuracy while maintaining the same compression ratio. With the help of model parameters pruning,

3) Provides an energy optimization approach, and we investigated the effectiveness of pruning different proportion model parameters on the reconstruction accuracy of the Stacked-CAE model when running at the same compression ratio as the model.

The following is the structure of the remainder of this paper: It is explained in depth in Section 2 about the design of the Stacked CAE-AE as well as the specifics of Stacked CAE training. The findings of the experiments are discussed in Section 3. In particular, we employ a number of techniques for energy optimization. Section 4 presents a summary of our method and discusses potential future work.

II. STACKED AE MODEL

A. Architecture of the Stacked CAE

The Stacked-CAE model contains two parts. The first part of the Stacked CAE-AE model we call encoder **E**, which includes four standard RBMs (in Figures 1 and 2).

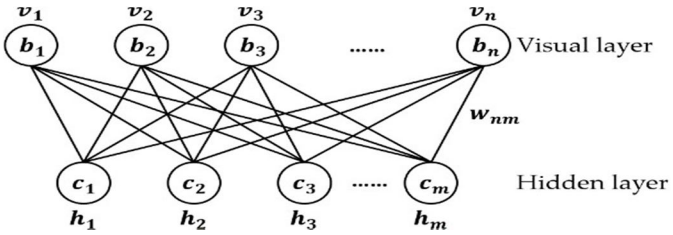


Figure 1. Standard RBM.

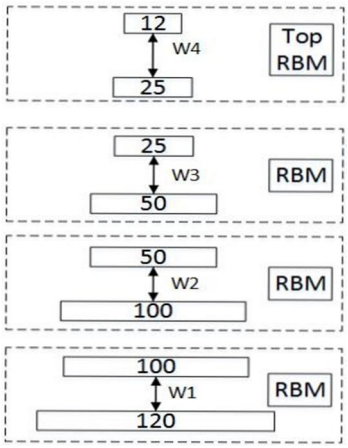


Figure 2. Stacked CAE.

Each standard RBM is an undirected graph model, which includes two layers. The vector **v** (v_1, v_2, . . . , v_n) represents the visual layer, which is the input vector with *n* scalar components from the training dataset; The vector **h** (h_1, h_2, . . . , h_m) represents the hidden layer vector with m scalar components. The standard RBM takes the state space (**v**, **h**) ∈ {0, 1} $^{m+n}$. The goal of RBM is to find P_{data} (**v**) the unknown true high dimensional distribution of the visual layer variables [30]. To achieve this goal, the high-dimensional distribution of the training dataset is modelled to get $P(\mathbf{v}|\theta)$, where the sample distribution depends on the model parameters θ [31].

For the second part of the Stacked CAE-AE model, firstly we use the characteristics of auto-encoder to flip the encoder **E** to get a symmetrical scale decoding output named decoder **D**; the structure is shown in Figure 3. The initial input of the encoder **E** is used as the input of the multilayer auto-encoder. Our goal is to satisfy the top output approximately equal to the bottom input.

B. Details of Stacked CAE Training

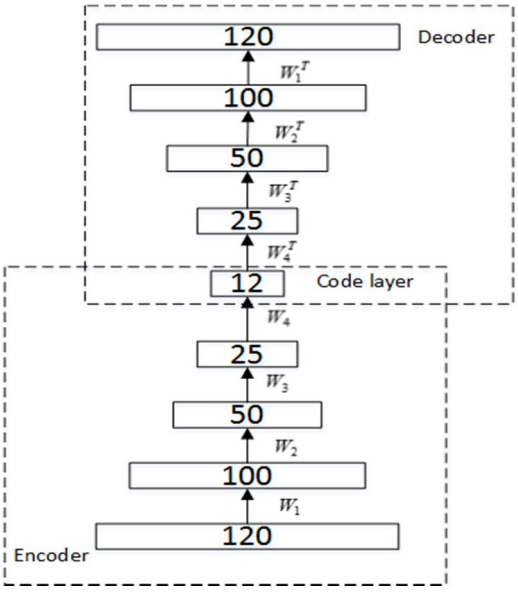


Figure 3. Stacked CAE.

III. RESULTS AND DISCUSSION

A. Preprocessing Dataset

The experimental data was collected by the Intel Lab wireless sensor network research team from the University of California, who placed 54 sensors to collect environmental data information in thelaboratory from 28 February to 5 April 2004.

In order to reduce the difference between the input data of the compressed model, and converge the algorithm more quickly, we mapped the original data to [0, 1] by *max* − *min* normalization, where *max* is the maximum value of the node sensing data, *min* is minimum value.

After the pre-processing is completed, the node temperature database is converted to a data set. Each node collects temperature data as a column vector, and stores this data in the temperature.txt file. Due to node failure, some nodes only recorded a small amount of data. The average number of data by each node after pre-processing is 29,665. The maximum number of data in all nodes is 55,080, and the minimum number is 2507. We divide each node data into two parts: The training set and the test set. In order to ensure that each node has enough training data to train the model, we did not use common split standards such as 7:3 and 6:2:2, and instead used 9:1.

The more the number of samples in the training set, and the closer the empirical distribution is to the true distribution, the more accurate will the fitness of model distribution be. Increasing the number of training sets is an effective measure to prevent over-fitting. We provide following three methods: (1) Obtain data from the source; (2) estimate the data distribution parameters of the training set by statistical methods such as point estimation and interval estimation, and use this distribution to generate more data; (3) augment the training set by interpolation in the original training set, such as Kriging Interpolation and Natural Neighbor Interpolation.

B. Compression and Reconstruction

We used the following performance criteria to evaluate the performance of our compression algorithm: (1) compression ratio (*CR*); (2) percentage RMS difference (*PRD*) [26]; (3) quality-score (*QS*) [26]. The definitions and formulas of these performance criteria are as follows:

Compression Ratio (*CR*): It is defined as the compressed data length over the size of uncompressed data, as shown in Equation (6).

$$CR = DR/D_{or} * D_{Cp} \quad (6)$$

Where D_{or} is the number of bytes of all original data. D_{cp} is the number of bytes of all compressed data. The *CR* value is expected to be high for an effective compression algorithm.

Percentage RMS Difference (*PRD*): It is a widely used performance measure that is used for calculating the quality of reconstructed data in the compression. The *PRD* value is expected to be aslow as possible for a quality compression approach.

Quality-Score (*QS*): Other important evaluation criteria for determining the effectiveness of compression algorithms is the *QS* value. *QS* is the ratio of *CR* to *PRD*. It represents the reconstructed data quality. The larger the *QS* value, the better the compression and reconstruction performance.

$$QS = CR/PRD \quad (7)$$

The learning rate determines how far the weights move in the gradient direction in a mini-batch, which is usually set by the experimenter. If the learning rate is small, the training will become more reliable, but optimization will take a long time, because each step towards the minimum of the loss function is small. If the learning rate is big, the training may not converge and could also diverge. Optimization will cross the minimum value and cause loss function to become worse. There are many ways to set an initial value for the learning rate. A simple solution is to try a few different values to see which value will optimize the loss function without loss in training speed. In the pre-training phase, we refer to the parameters in Ref. [36], set the learning rate of 0.01/batch-size (the batch-size is 120). In the retraining phase, we start with a value of 0.1, then exponentially reduce the learning rate to 0.01, 0.001 and 0.0001. In order to find the optimal learning rate, we set different learning rates to train the model and record the summation of loss value when the model loss is no longer reduced [37]. Figure 5 shows the loss value of a model with different learning rates. We sought to find a point with the smallest value of model loss. In our experiment, we found that when the learning rate was between 0.0001 and 0.001, the model loss value was the smallest. We finally selected a learning rate of 0.0001.

We first use the training set of node 7 to pre-train the model without retraining, and test the efficiency of a different number of pre-training iterations to the compression performance of the model. Then we use the test set of node 7 to calculate the *PRD* value and the *QS* value of the model under different numbers of pre-training iterations. Similarly, we use the training set of node 7 to retrain the model with a fixed number of pre-training iterations. We use the test set of node 7 to calculate the *PRD* value and the *QS* value of the model under different numbers of retraining iterations. During the test, we added the *PRD* value and the *QS* value of each mini-batch of the test set, and then averaged the sum value as the final result. The number of mini-batches of the test set is 325. The results are shown in Figure 6.

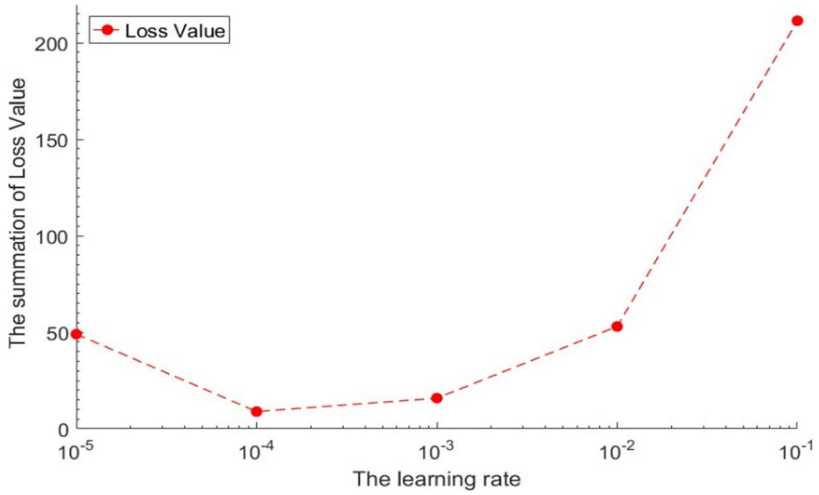
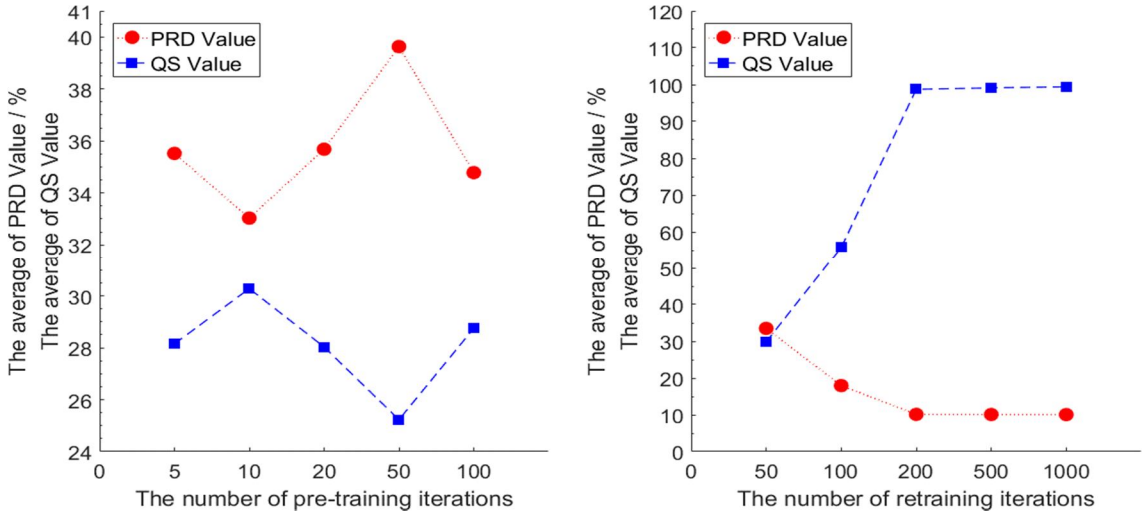


Figure 5. The loss value of model with different learning rates.

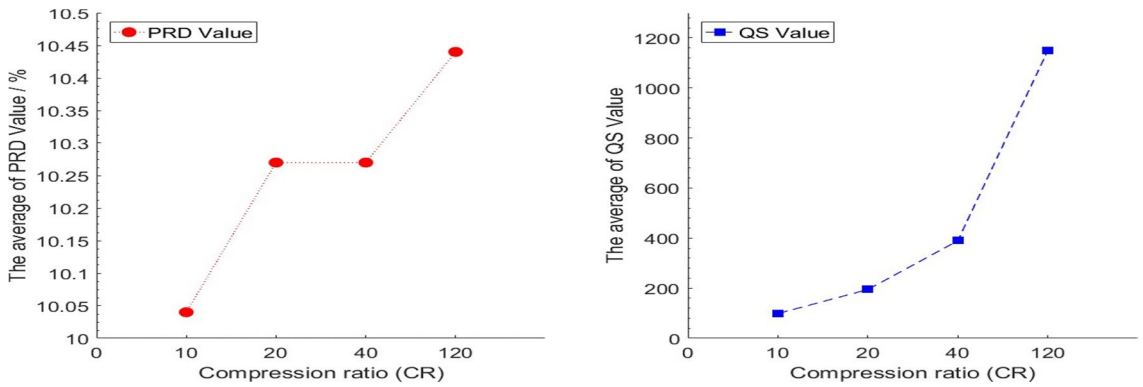


(a) (**b**)

Figure 6. (a) Compression performance under different number of pre-training iterations without retraining; (b) compression performance under different retraining iterations with the number of pre-training iterations as 10.

Without retraining, the compression performance of the model will not increase and could even decrease with an increase in the number of pre-training iterations. When the number of pre-training iterations is 10 and the number of retraining iterations is 0, the compression performance of the model is optimal with the smallest *PRD* value and the largest *QS* value. Thus, we set the number of pre-training iterations of our model to 10. When the number of pre-training iterations is fixed, increasing the number of retraining iterations can significantly improve the compression performance of the model. When the number of retraining iterations reaches 200 and above, the compression performance of the model tends to be stable. At this time, increasing the number of retraining iterations does not significantly improve the compression performance of the model. Considering that the model needs to be calculated on the sensor node, increasing the number of retraining iterations will lead to an increase in calculation energy consumption. We finally selected the number of retraining iterations as 200.

In this experiment, we test the compression performance of the model under different *CRs*. This experiment data uses the data of node 7 with the number of pre-training iterations of 10, and the number of retraining iterations of 200. We then set *CR* as 10, 20, 40, and 120. We use the test set of node 7 to calculate the *PRD* value and the *QS* value of the model under different *CRs*. During the test, we summed up the *PRD* value and the *QS* value of each mini-batch of the test set, and then averaged the sum value as the final result. The number of mini-batches of the test set is 325. Figure 7 shows the compression performance of the model under different *CRs*.



(**a**) (**b**)

Figure 7. (a) The *PRD* value under different *CRs*, (b) The *QS* value under different *CRs*.

Figure 7 illustrated that for a single node, with the increase of the *CR*, the compression performance of the model is not significantly increased or decreased, which shows that the model can imbibe the inherent properties of the data. These inherent properties are inherently weighted on the weight matrix and are independent of the dimension after compression. This is the difference between the deep compression method and the shallow compression method, since the reconstruction error of the shallow compression method increases with an increase in the *CR*. The result means that our algorithm can get a higher *CR* in the case of minimal reconstruction error. In our experiments, when *CR* was 10, the *PRD* value was the smallest. The reconstructed data value is closest to the original data value. Although increasing *CR* can significantly increase the *QS* value, the *PRD* value will also increase, which represents a difference between the reconstructed data, and the original data becomes larger. In this experiment, we explore the optimal compression performance of the model, and in the next experiment, we set the *CR* to 10 to explore the reconstruction performance of the model.

Figure 8 shows the reconstructed data and the original data of node 7 for our model, with the number of pre-training iterations being 10, the number of retraining iterations being 200, and *CR* of 10. We first use the model to compress the original data, and then use the model to reconstruct the compressed data. For all the data in the test set of node 7, we sum up the absolute value error between the original data and the reconstructed data, and then average the sum value as the final result. The average absolute value error obtained was 0.2815 ˚C, maximum value was 0.4602 ˚C, and minimum value was 0.0026 ˚C. Our model has been proven to have higher reconstruction accuracy, and the reconstructed data can correctly approximate the trend and value of the original data.

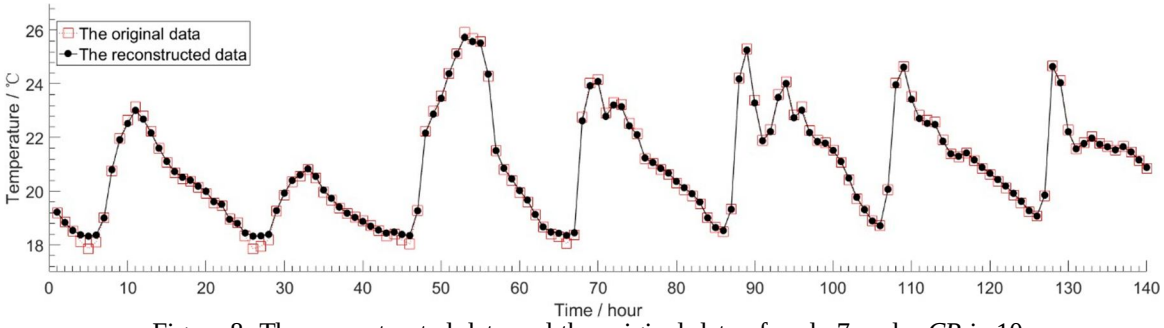


Figure 8. The reconstructed data and the original data of node 7 under *CR* is 10.

In the following experiments, we compare the performance of our algorithm with other compression algorithms. Performing CS algorithm on 40,000 data points for node 7. Since the length of stream data for CS algorithm cannot be too long, we divide these points into eight segments with segment length of 5000. We average the results of all segments as the final result of CS algorithm, and set the *CR* at 10. The results are shown in Table 1. At the same time, we test the performance of our algorithm on different data sets. The results are shown in Table 2.

Table 1. Compression performance comparison.

Algorithm	*PRD* (%)	*QS*	Reconstruction Data Error (˚C)
CS	38.40	26.04	1.4143
Standard RBM	33.03	30.28	1.0423
Our algorithm	10.04	99.60	0.2815

Table 2. Model performance on other datasets.

Dataset	*PRD* (%)	*QS*	Reconstruction Data Error
Argo (temperature)	11.10	90.09	0.8434 (˚C)
ZebraNet (location/UTM format)	9.82	101.83	259.26
CRAWDAD (speed)	8.53	117.23	6.2056 (km/h)
Intel Lab (humidity)	10.90	91.74	3.8383 (%RH)

IV. CONCLUSION

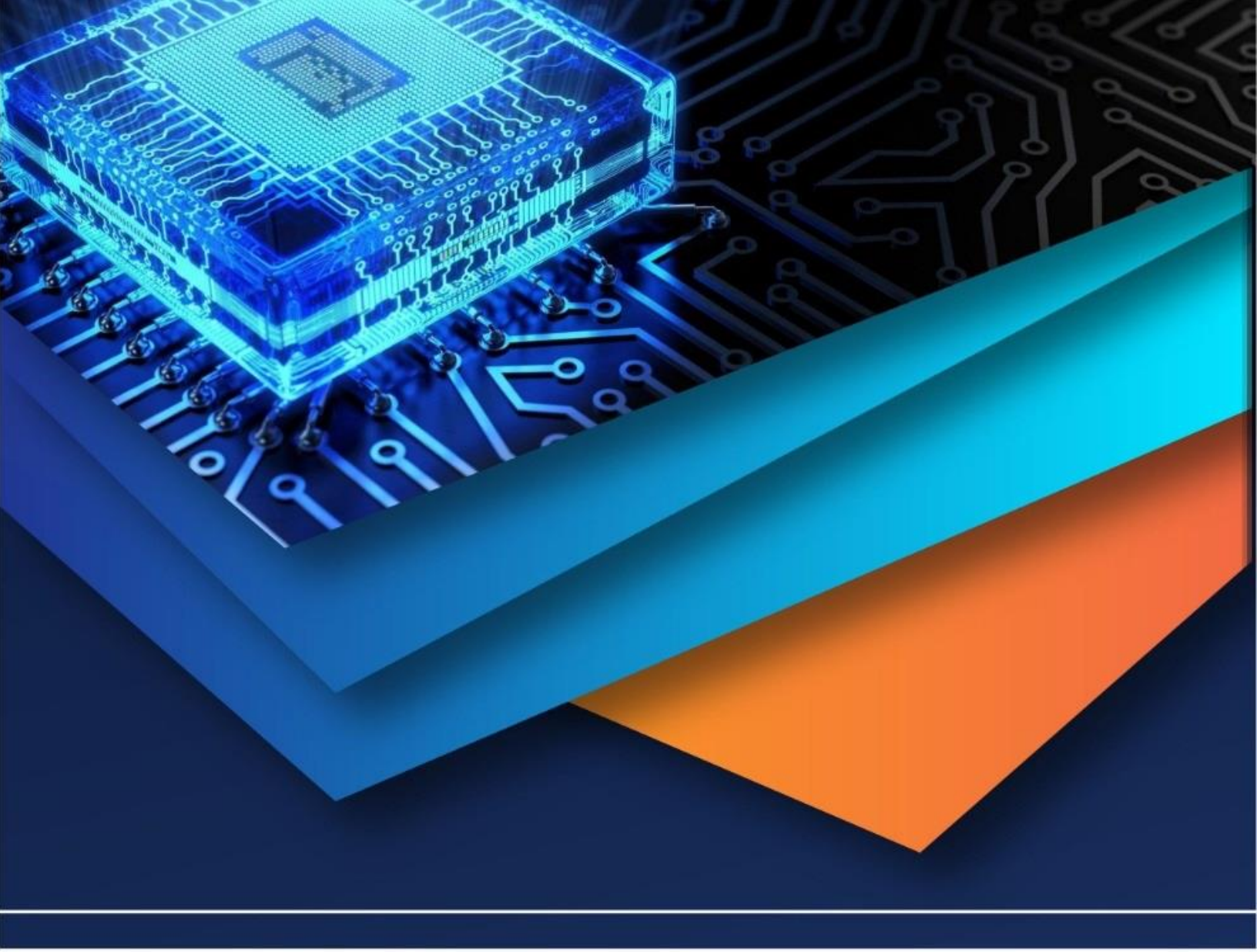
Conclusions In this research, we present a Stacked CAE model for compressing WSNs data employing Convolutional RBM and AE in conjunction with each other. We suggest a model parameter adjustment technique that is divided into two parts: pre-training and retraining, with the goal of improving compression performance as a result. Experiment studies are used to determine the effectiveness of the number of iterations of pre-training and retraining on the performance of the model. We also provide a method for applying the model to wireless sensor networks (WSNs) and examine the computational efficiency of the approach. We apply the strategy of prune parameters to further improve the energy consumption of our algorithm, taking into account the calculation and communication energy consumption. Our experimental results demonstrate that our algorithm has superior transfer learning ability as well as superior reconstruction accuracy when compared to traditional algorithms when operated under the identical CR. It is possible to cut the energy usage of data communication by 90 percent. The sensor node is typically outfitted with a number of sensors that collect data from a variety of environmental monitoring devices. Joint compression and reconstruction of multi-stream data can be accomplished using this technology, which can be enhanced. It is inevitable that the theoretical system error will be incorporated into this method because Gibbs sampling and k-step divergence are utilized to estimate the probability distribution of reconstructed data in this method. Exploring ways to reduce the systematic inaccuracy of the model will be the subject of our next research project, which will be completed this year. Deep learning models such as VAE, GAN, and other mixed deep learning models can also be used.

REFERENCES

[1] Lazarescu, M.T. Design of a WSN platform for long-term environmental monitoring for IoT applications. J. Emerg. Sel. Top. Circuits Syst. 2013, 3, 45–54. [CrossRef]
[2] Janai, J.; Güney, F.; Behl, A.; Geiger, A. Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. arXiv 2017, arXiv:1704.05519.
[3] Kuo, T.W.; Lin, K.C.J.; Tsai, M.J. On the Construction of Data Aggregation Tree with Minimum Energy Cost in Wireless Sensor Networks: NP-Completeness and Approximation Algorithms. IEEE Trans. Comput. 2016, 65, 3109–3121. [CrossRef]
[4] Mcdonald, D.; Sanchez, S.; Madria, S.; Ercal, F. A survey of methods for finding outliers in wireless sensor networks. J. Netw. Syst. Manag. 2015, 23, 163–182. [CrossRef]
[5] He, S.; Chen, J.; Yau, D.K.Y.; Sun, Y. Cross-Layer Optimization of Correlated Data Gathering in Wireless Sensor Networks. IEEE Trans. Mob. Comput. 2012, 11, 1678–1691. [CrossRef]
[6] Keogh, E.; Chakrabarti, K.; Pazzani, M.; Mehrotra, S. Locally adaptive dimensionality reduction for indexing large time series databases. ACM Sigmod Rec. 2001, 30, 151–162. [CrossRef]
[7] Buragohain, C.; Shrivastava, N.; Suri, S. Space efficient streaming algorithms for the maximum error histogram. In Proceedings of the IEEE 23rd International Conference on Data Engineering, Istanbul, Turkey, 16–20 April 2007; pp. 1026–1035.
[8] Li, M.; Lin, H.J. Design and Implementation of Smart Home Control Systems Based on Wireless Sensor Networks and Power Line Communications. IEEE Trans. Ind. Electron. 2015, 62, 4430–4442. [CrossRef]
[9] Donoho, D.L. Compressed sensing. IEEE Trans. Inf. Theory 2006, 52, 1289–1306. [CrossRef]
[10] Eldar, Y.C.; Kutyniok, G. Compressed Sensing: Theory and Applications; Cambridge University Press: Cambridge, UK, 2012; pp. 1289–1306.
[11] Candès, E.J.; Wakin, M.B. An introduction to compressive sampling. IEEE Signal Process. Mag. 2008, 25, 21–30. [CrossRef]
[12] Zhang, Z.; Xu, Y.; Yang, J.; Li, X.; Zhang, D. A survey of sparse representation: Algorithms and applications. IEEE Access 2015, 3, 490–530. [CrossRef]
[13] Haupt, J.; Bajwa, W.U.; Rabbat, M.; Nowak, R. Compressed sensing for networked data. IEEE Signal Process. Mag. 2008, 25, 92–101. [CrossRef]
[14] Caione, C.; Brunelli, D.; Benini, L. Distributed Compressive Sampling for Lifetime Optimization in Dense Wireless Sensor Networks. IEEE Trans. Ind. Inform. 2012, 8, 30–40. [CrossRef]
[15] Ranieri, J.; Rovatti, R.; Setti, G. Compressive sensing of localized signals: Application to analog-to-information conversion. In Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS), Paris, France, 30 May–2 June 2010; pp. 3513–3516.
[16] Brunelli, D.; Caione, C. Sparse recovery optimization in wireless sensor networks with a sub-nyquist sampling rate. Sensors 2015, 15, 16654–16673. [CrossRef] [PubMed]
[17] Xiang, L.; Luo, J.; Vasilakos, A. Compressed data aggregation for energy efficient wireless sensor networks. In Proceedings of the 2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), Salt Lake City, UT, USA, 27–30 June 2011; pp. 46–54.
[18] Li, S.; Xu, L.D.; Wang, X. Compressed sensing signal and data acquisition in wireless sensor networks and internet of things. IEEE Trans. Ind. Inform. 2013, 9, 2177–2186. [CrossRef]
[19] Wu, M.; Tan, L.; Xiong, N. Data prediction, compression, and recovery in clustered wireless sensor networks for environmental monitoring applications. Inf. Sci. 2016, 329, 800–818. [CrossRef]
[20] Sheltami, T.; Musaddiq, M.; Shakshuki, E. Data Compression Techniques in Wireless Sensor Networks. Future Gener. Comput. Syst. 2016, 64, 151–162. [CrossRef]
[21] Bhosale, R.B.; Jagtap, R.R. Data Compression Algorithm for Wireless Sensor Network. Int. Res. J. Multidiscip. Stud. 2016, 2, 1–6.
[22] Ying, B. An energy-efficient compression algorithm for spatial data in wireless sensor networks. In Proceedings of the 18th IEEE International Conference on Advanced Communications Technology, PyeongChang, Korea, 31 January–3 February 2016; pp. 161–164.
[23] Hinton, G.E.; Salakhutdinov, R.R. A better way to pretrain deep boltzmann machines. In Proceedings of the Twenty-Sixth Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–8 December 2012; pp. 2447–2455.

[24] Tramel, E.W.; Manoel, A.; Caltagirone, F.; Gabrié, M.; Krzakala, F. Inferring sparsity: Compressed sensing using generalized restricted Boltzmann machines. In Proceedings of the 2016 IEEE Information Theory Workshop (ITW), Cambridge, UK, 11–14 September 2016; pp. 265–269.
[25] Papa, J.P.; Rosa, G.H.; Marana, A.N.; Scheirer, W.; Cox, D.D. Model selection for Discriminative Restricted Boltzmann Machines through meta-heuristic techniques. J. Comput. Sci. 2015, 9, 14–18. [CrossRef]
[26] Tomczak, J.M.; Zięba, M. Classification Restricted Boltzmann Machine for comprehensible credit scoring model. Expert Syst. Appl. 2015, 42, 1789–1796. [CrossRef]
[27] Carreira-Perpinan, M.A.; Hinton, G.E. On contrastive divergence learning. Aistats 2005, 10, 33–40.
[28] Tulder, G.V.; Bruijne, M.D. Combining Generative and Discriminative Representation Learning for Lung CT Analysis with Convolutional Restricted Boltzmann Machines. IEEE Trans. Med. Imaging 2016, 35, 1262–1272. [CrossRef]
[29] Côté, M.A.; Larochelle, H. An Infinite Restricted Boltzmann Machine. Neural Comput. 2016, 28, 1265–1288. [CrossRef] [PubMed]
[30] Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. Nature 1986, 323, 533–536. [CrossRef]
[31] Salakhutdinov, R.; Mnih, A.; Hinton, G. Restricted Boltzmann machines for collaborative filtering. In Proceedings of the 24th International Conference on Machine Learning, Corvalis, OR, USA, 20–24 June 2007; pp. 791–798.
[32] Smith, L.N. Cyclical learning rates for training neural networks. In Proceedings of the 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), Santa Rosa, CA, USA, 24–31 March 2017; pp. 464–472.
[33] Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; Graf, H.P. Pruning filters for efficient convnets. arXiv 2016, arXiv:1608.08710.
[34] Han, S.; Pool, J.; Tran, J.; Dally, W. Learning both weights and connections for efficient neural network. In Proceedings of the Twenty-Ninth Conference on Neural Information Processing Systems, Montréal, QC, Canada, 7–12 December 2015; pp. 1135–1143.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)