Design of various Image Compression Methods in Wireless Sensor Networks

S. Aruna Deepthi, E.Sreenivasa Rao, M.N.Giriprasad

Abstract: ***The processing capacity and power of nodes in a Wireless Sensor Network (WSN) are restricted. The quality of the images is deficient, and the contents of the images may vary after decoding when we apply image compression algorithms in WSN. Various compression algorithms are compared in this paper. An Image Compression method based on Restricted Boltzmann Machine (RBM), Auto encoders and Non-negative Matrix Factorization (NMF), Least Square Non-Negative Matrix Factorization (LSNMF), Projective Non-Negative Matrix Factorization (PNMF) network are proposed in this paper. For the WSN, we have used a Message Queue Telemetry Transport (MQTT) protocol. We have used a three Raspberry Pi's to build a WSN; Publisher, Broker, Subscriber. A Publisher, where it can trigger the camera and captures the images then compress it and send it to another raspberry pi which is a MQTT broker. The PSNR values for those image compression methods were analyzed and compared against each other for images evaluated from the MNIST dataset. Along with the simulation results, all these compression methods are implemented using hardware implementation. Raspberry Pi, a single-board computer with in-built Wi-Fi capabilities, was used in establishing a WSN. Message Queue Telemetry Transport (MQTT) protocol was used for transmitting the compressed images across the WSN, that offers fast and reliable transmission.***

Keywords: Auto Encoder, RBM, NMF, PNMF, LSNMF, WSN, MQTT

I. INTRODUCTION

Wireless Sensor Nodes (WSNs) play a very significant role in our day to day applications. As the resources in each sensor node are limited. It is challenging to reduce energy consumption and increase the lifetime of a sensor node.

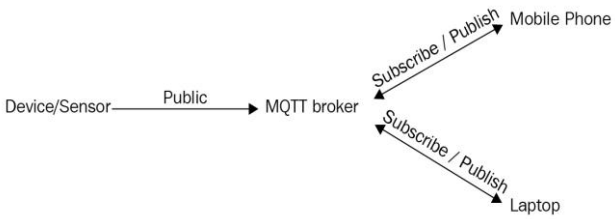

Figure 1. MQTT Architecture

Currently, the Image Compression algorithms in WSNs are subject to the random changes in image contents. It is difficult to describe various images in the real world with only one kind of image model. The neural network model is adopted in WSNs to compress the images. We have used three image compression algorithms in this work for implementing the WSN. They are Autoencoders, Restricted Boltzmann Machines (RBM), and Non-Negative Matrix Factorization (NMF).

An autoencoder learns to compress data into a low-dimensional representation and then reconstructs that representation into something that matches the original data. This way, the low-dimensional description ignores the noise, which does not help rebuild the original data.

There are various types of learning in Machine learning, supervised learning, unsupervised learning, and reinforcement learning. The example of unattended Restricted Boltzmann machines (RBMs) is an unsupervised class of machine learning algorithms. These algorithms are used to represent the internal representation of data. This allows the RBM to sample the output of the visible units independently, given the hidden layer input, and vice versa.

The examples of low-rank approximations are canonical methods, such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Independent Component Analysis (ICA), Vector Quantization (VQ), etc., they differ from one

Fig: 2: Raspberry Pi Single-board Computer

Another in the statistical properties attributable to the different constraints imposed on the component matrices and their underlying structures. Paatero and Tapper first introduced NMF as the concept of Positive Matrix Factorization, which concentrated on a specific application with Byzantine algorithms.

LSNMF gives patterns analysis of microarray datasets. LSNMF is better than NMF as it introduces uncertainty

Revised Manuscript Received on October 05, 2019

S. Aruna Deepthi, Assistant Professor, Department of E.C.E J.N.T.U.A., Anantapuramu, India.

E.Sreenivasa Rao, Professor and HoD, Department of E.C.E, Vasavi college of Engineering, Hyderabad, India.

M.N.Giriprasad, Professor, Department of E.C.E, J.N.T.U. A, Anantapuramu, India,

estimates in update rules. Due to this reason LSNMF is more stable than NMF datasets, noise is less and sensitivity to signals is more.

For transmitting images over the WSN network, we have used MQTT Protocol. MQTT is a lightweight open messaging standard maintained by OASIS. MQTT is used whenever extensive data has to be collected in real-time applications like weather, telematics, and transportation.TCP /IP connection is maintained between the clients and a broker server.

. Clients subscribe to messages of interest. The MQTT broker uses this connection to push new messages to clients who have subscribed to those messages.

For resource-constrained scenarios, such as a WSN, maintaining TCP/IP connections could be too complicated for low-footprint sensor-actuator (SA) nodes.

The Raspberry Pi is known as a single-board computer, which is a computer, desktop, laptop, or smartphone, but built on a single printed circuit board. Like most single-board computers, the Raspberry Pi is small like a credit card. It runs on Raspberry pie, a Linux based Operating system, thereby making it easier for the developer to run software like on a Windows-based computer.

In building our WSN network, we use multiple Raspberry Pi boards that can perform a particular function in the WSN.In this one unit can act as a broker to establish the communication with other Raspberry Pi units in the network, while the other groups can publish (send) or subscribe (receive) images across the WSN network. The broker handles the scheduling, transmission, and authorization of the publishers/subscribers in the WSN.

II. RELATED WORK

The main differences [1] lie in the way is the way we deal with compress the images and transfer the images through WSN. But they haven't used an MQTT protocol which is fast and reliable. They have used only RBM technique; we used three compression algorithms and transferred images through a wireless sensor network.

Theis, Lucas [2] proposed a method to compress the images using an autoencoder. The technique they have introduced results in the lossy images.

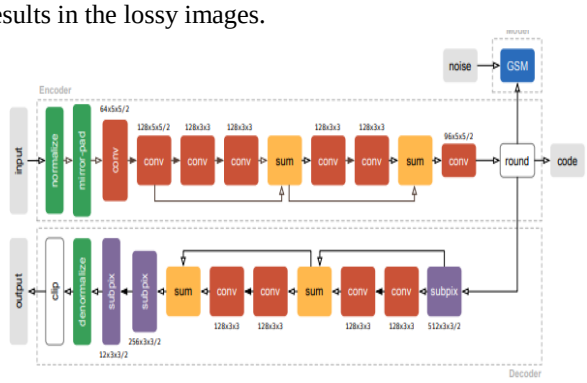

Fig 3: Compressive Autoencoder Architecture

Younghoon (2018) proposed a method based on wireless sensor networks. They are capturing images from a camera sensor and transmit via a wireless sensor network. They are not using any compression algorithm before transferring the data. This is resulting in slow transfer rates.

Singh, Ramnik, and Anil Kumar Verma in (2017) proposed a way of image transfer via wireless sensor networks. They have used a cross-layer optimization technique to transfer the images throughout the network. But they have a limited capability of sensor nodes make it difficult to transmit the images as they require substantial amounts of overheads to be sent.

Atmoko, R. A., R. Riantini, and M. K. Hasin in 2017 implemented a Real-time data acquisition using MQTT. They have transferred the real-time text data though out the wireless network. The main difference with our project is not dealing with image transmission.

Ahmed, Sevil, Andon Topalov, and Nikola Shakev in 2017 implemented [32] a robust based wireless sensor network using MQTT protocol. They have used cloud computing technology to communicate. But we are using hardware which can transmit communicate wirelessly.

Jaladi, Aarti Rao, Karishma Khithani, Pankaja Pawar, Kiran Malvi, and Gauri Sahoo in 2017 implemented a Wireless sensor network[22] based on IoT. The difference between our project lies in hardware and software they have used. They have used a ZigBee module to transmit wirelessly. The limitation here is ZigBee can transfer images in a 2.4kb/s. It is very slow than our hardware. Another hardware is a Bluetooth module which is used to transmit via Bluetooth. This could be another limitation.

III. IMPLEMENTATION OF IMAGE COMPRESSION METHODS

A. *Autoencoders*

The autoencoder neural network is an example of unsupervised algorithm. It is applied to the back propagation and the target values to be equal to the inputs y(i) = x(i).

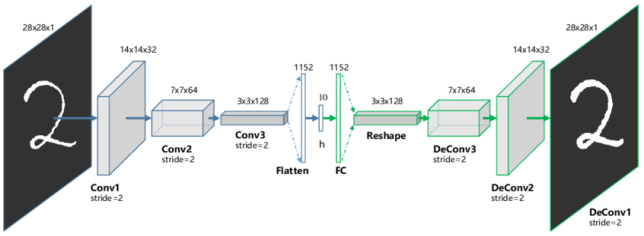

Fig 4: Convolutional Autoencoders.

To achieve the desired output, features will be compressed during the middle layer, as the convergent and divergent layers are used.

A sample has been shown. The input and output layers have five neurons, whereas the number of neurons has been gradually decreased in the middle sections. The compressed layer has only two neurons, which is the number of latent dimensions we would like to extract from the data.

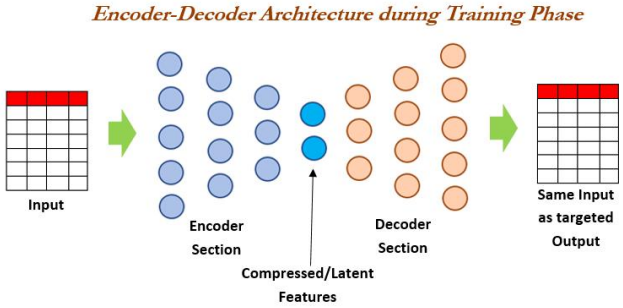

Fig 5: Encoder-Decoder Architecture during the Training Phase

To define the encoder () and the decoder (), an autoencoder can be mathematically represented as follows:

$$\phi : \mathcal{X} \rightarrow \mathcal{F}$$
$$\psi : \mathcal{F} \rightarrow \mathcal{X}$$
$$\phi, \psi = \underset{\phi,\psi}{\arg\min} \|X - (\psi \circ \phi)X\|^2$$

[3]

As mentioned in the equation, the parameters of the encoder and the decoder are optimized in a way that minimizes a special kind of error, which is known as reconstruction error. Reconstruction error is the error between the reconstructed input and the original input.

1) Auto encoder Train Parameters:

The following parameters may be set to train function of convolutional autoencoder.

Number of batches: The training set is divided into this many batches for stochastic gradient descent Here batch size is 100 taken into consideration.

Learning rate: This is the initial learning rate. This should be small. We have taken learning rate as 0.01.

Max epochs: This is a backstop, in other words, insurance against endless iteration. A maximum epoch 2000 is taken for consideration.

2) Compression Algorithm Implementation in Python:

The whole image compression algorithm is implemented in python with TensorFlow backend that supports the neural network specific libraries.

i. Read and load a MNIST Dataset.
ii. Extract the images from the dataset and load into a NumPy array.
iii. Pre-processing the image into default size which is 28*28 and color mode is grayscale.
iv. Creating the Neural Network model for encoding the Image dataset.
v. Creating the Neural Network model for decoding the compressed image dataset.
vi. Train the model using train parameters as we discussed earlier.
vii. Validate the test images and compute the loss.
viii. Save the model and load the model for testing new images.
ix. Calculate the PSNR ratio for original image and compressed images.

B. Restricted Boltzmann Machines

Boltzmann machines of the unrestricted type consist of a neural network with an input layer and several hidden layers. The neurons or units in the neural network make stochastic decisions about whether to turn on or not based on the data fed in during training and the cost function the Boltzmann machine is trying to minimize. With this training, the Boltzmann machine discovers interesting features about the data, which helps model the complex underlying relationships and patterns present in the data.

The following diagram illustrates the architecture of RBM.

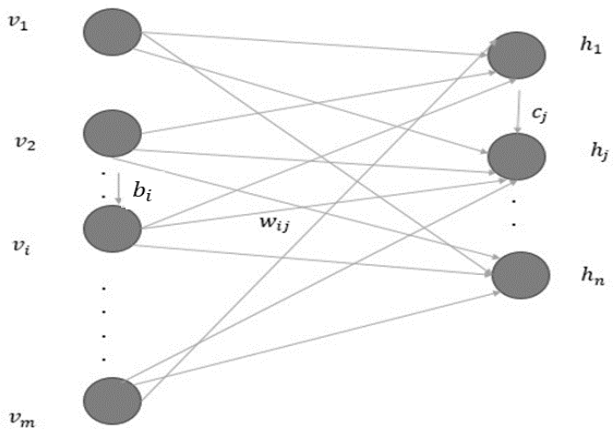

Fig 6: Restricted Boltzmann Machines (RBM) Architecture

The weight, wij ∋ W, connects the visible unit, i, to the hidden unit, j, where W ∋ Rm x n is the set of all such weights, from visible units to hidden units. The biases in the visible units are represented by bi ∋ b, whereas the biases in the hidden units are represented by cj ∋ c.

Inspired by[17] ideas from the Boltzmann distribution in statistical physics, the joint distribution of a visible layer vector, v, and a hidden layer vector, h, is made proportional to the exponential of the negative energy of the configuration:

$$P(v, h) \propto e^{-E(v,h)} \quad [4]$$

Before we compress the images using RBM, let's first build one using matrix factorization, one of the most successful and popular collaborative filtering algorithms today. Matrix factorization decomposes the user-item matrix into a product of two lower dimensionality matrices. Users are represented in lower dimensional latent space, and so are the items.

Assume our user-item matrix is R, with m users and n items. Matrix factorization will create two lower dimensionality matrices,

H and W. H is an "m users" x "k latent factors" matrix, and W is a "k latent factors" x "n items" matrix.

The ratings are computed by matrix multiplication:

R = H__W. [5]

The number of k latent factors determines the capacity of the model. The higher the k, the greater the capacity of the model. By increasing k, we can improve the personalization of rating predictions for users, but, if k is too high, the model will overfit the data.

1) RBM Train Parameters:

All parameters are pre-set to defaults that should be reasonable for many or most applications. The following parameters may be set:

Random initialization iterations: This is the number of trial weight sets that are tested to find a good starting point for stochastic gradient descent training. Here we have used 2000.

Number of batches: The training set is divided into this many batches for stochastic gradient descent Here batch size is 100 taken as consideration.

Learning rate: This is the initial learning rate. This should be small, probably smaller than the value the user is accustomed to for other programs. We have taken learning rate as 0.5.

Max epochs: This is a backstop, in other words, insurance against endless iteration. A maximum epoch 100 is taken for consideration.

2) Compression Algorithm Implementation in Python:

The whole image compression algorithm is implemented in python with TensorFlow backend that supports the neural network specific libraries.

i) Read a MNIST dataset: Download a dataset which are archives from a MNIST website.
ii) Read MNIST image - Unzip the dataset and read all images into a NumPy array.
iii) Split the dataset for test and train - Splitting of a dataset for train set of images to train the model and test set of images is to validate the model.
iv) Adding the Optimization function - Optimization function is to generate the loss between train real images and predicted images.
v) CNN neural network model for Image Compression: A Neural network model is to create the network between input and output.
vi) Train the model using train parameter as we have discussed earlier.
vii) Predict the compressed images.
viii) Validate the test images and compute loss.
ix) Save the model in a local directory.
x) Load the model and test it for new images.
xi) Calculate the PSNR Ratio for original image and compressed image.

A standard MNIST-format image file is read.

Non-Negative Matrix Factorization

When the dataset, X, is non-negative, it is possible to apply a factorization technique, which has been proven to be more reliable when the goal of the task is to extract atoms that correspond to the structural parts of the samples. For example, in the case of images, they are supposed to be geometrical elements or even more complex parts. The main condition imposed by Non-Negative Matrix Factorization (NMF) is that all of the matrices involved must be non-negative and X = UV. Hence, once a norm, N, has been defined, the simple objective becomes the following:

[6]

As this is not generally acceptable when sparsity is also

$$min_{U,V} \|X - UV\|_N$$

needed (and, moreover, to allow for greater flexibility in changing the solution to meet specific requirements), the problem is often expressed (such as in scikit-learn) by adding penalties on both Frobenius (a matrix extension of L2) and L1 norms (for example, in ElasticNet):

$$min_{U,V} \left[\|X - UV\|_N + \alpha\beta (\|U\|_1 + \|V\|_1) + \frac{1}{2}\alpha(1-\beta) \left(\|U\|^2_{Frob} + \|V\|^2_{Frob} \right) \right]$$

[7]

Double regularization allows you to obtain both sparsity and a better match between the components and the parts of the samples, by avoiding an effect similar to the over fitting of supervised models (as the solution is sub-optimal, it also more flexible in adapting to new samples drawn from the same data-generating process; this increases the likelihood generally achievable).

Let's consider the MNIST dataset to take into account for compressing the images. Three components are dominant (3, 24, and 45); hence, we can try to express the sample as a combination of them. The coefficients are, respectively, 0.19, 0.18, and 0.16. The result is shown below in fig 7 (the digit X [0] is the representation of zero):

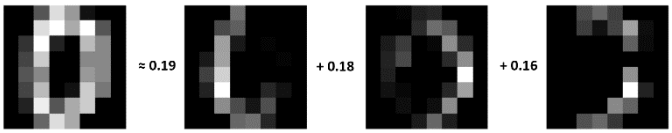

Fig 7: Result of the step by step compression of images using Non-Negative Matrix Factorization

1) NIMFA Library:

NIMFA is a Python library for NMF which includes implementations of several factorization methods, initialization approaches, and quality scoring. Both dense and sparse matrix representations are supported.

$$P=WW^T \qquad [8]$$

2) *Compression Algorithm Implementation in Python:*

- Load the images as a NumPy array.
- Pre-processing the images for getting image standards like size constraints, grayscale or RGB, etc.

- Applying NMF Algorithm to the images by using 'fit' method in NIMFA Library.
- Resulting compressed image after applied NMF Algorithm.

C. Least Squares Non-Negative Matrix Factorization:

Least Squares Non-Negative Matrix Factorization is another approach for NMF by using an alternative least square method.

1) Algorithm for Least Square Nonnegative Matrix Factorization:

i. Initialize two non-negative matrices B, C;
ii. First, set matrix C as a constant value, solve the Non-Negative Least Squares Problem with matrix B. Record the calculating times used above as m so matrix B consists of *m* optimal solutions ultimately.
iii. Then, set matrix B as a constant value similarly, solve the Non-Negative Least Squares.
iv. Problem with matrix C. Record the calculating times used above as *n* so matrix C consists of *n* optimal solutions ultimately.
v. If matrix B, C meets the requirements of stopping Least Square algorithm, then finish the calculation. Else, turn to Step 2.

2) Python Implementation for LSNMF:

i. Load the images as a NumPy array.
ii. Pre-processing the images for getting image standards like size constraints, grayscale or RGB, etc.
iii. Applying LSNMF Algorithm to the images by using 'fit' method in NIMFA Library.
iv. Resulting compressed image after applied LSNMF Algorithm.

D. Projective Non-Negative Matrix Factorization:

The standard NMF is a linear model which allows linear correlation. In NMF, the two matrices[3] W and H contain a total of r× (m+n) free parameters. This gives a certain ambiguity to the problem. For example, consider the simplest possible case in which V and W are just column vectors and H = H is a scalar: obviously, there is an infinite number of solutions W= 1/H V with H arbitrary. To handle the nonlinear correlation, the polynomial NMF (PNMF) algorithm was proposed. The main idea of PNMF[18] is to first transform data into higher dimensional feature space[26] by using a polynomial kernel-induced nonlinear mapping and then perform decomposition in that feature space. Based on this, a novel method which we called Projective Non-negative Matrix Factorization (PNMF) as the solution to the following optimality problem.

$$D(A||B) = \sum_{i,j} (A_{ij} log \frac{A_{ij}}{Bij} - A_{ij} + B_{ij}) \quad [9]$$

1) Python Implementation for PNMF:

i. Load the images as a NumPy array.
ii. Pre-processing the images for getting image standards like size constraints, grayscale or RGB, etc.
iii. Applying PNMF Algorithm to the images by using 'fit' method in NIMFA Library.
iv. Resulting compressed image after applied PNMF Algorithm.

E. Implementing WSN using Raspberry Pi via MQTT:

For implementing the WSN, we have utilized an MQTT (Message Queuing Telemetry Transport) Protocol for transferring the images across the Raspberry Pi's connected in the network.

The MQTT[21] protocol is a good choice for wireless networks that experience varying levels of latency due to occasional bandwidth constraints or unreliable connections. Should the connection from a subscribing client to a broker get broken, the broker will buffer messages and push them out to the subscriber when it is back online. Should the connection from the publishing client to the broker be disconnected without notice, the broker can close the connection and send subscribers a cached message with instructions from the publisher.

Fig 8: Raspberry Pi based WSN using MQTT.

The above diagram shows the network architecture of MQTT. Here we are using three devices. Publisher; Broker; Subscriber. Publisher can act as a publisher and subscriber. So, we can transmit and receive data. As well as,[32] subscriber can transmit and receive the data. A broker is a middleware or like a router. It can receive the data from publisher and sends it to the requested subscriber. So, a publisher can transmit images to the broker and at a time, the subscriber requests the data from the broker. Once the request received to the broker, it can send those received images to the subscriber.

IV. RESULTS AND DISCUSSION

This experiment consists three parts: the performance analysis of compression methods; the compression results of all methods; analysis of WSN performance when all methods of compression images have transmitted through WSN.

4.1 *Performance Analysis of Compression Methods:*

The datasets of used in our study are the famous hand written digits database which is are of MNIST.

The MNIST Database consists of 50,000 groups of training samples and test samples. Each train and test group of samples consists of a grayscale images which is of 28*28 resolution.

Fig 9: Sample Images of MNIST Database

Peak Signal Noise Ratio is evaluated for the performance of a reconstructed image quality. For an Image with M*N Pixels,

$$\mathrm{PSNR} = 10\log\frac{255^2}{(1/MN)\sum_1^M\sum_1^N[x(i,j)-\hat{x}(i,j)]^2}$$

[10]

Where $x\ (i,\ j)$ and $x\ (i,\ j)\hat{}$, respectively, denote the gray values of the original image and the reconstructed image

4.2. *Compression Results:*

4.2.1 *Autoencoder*: The Proposed method was tested with 5 different numbers of test images. Here are the following original images and reconstructed images.

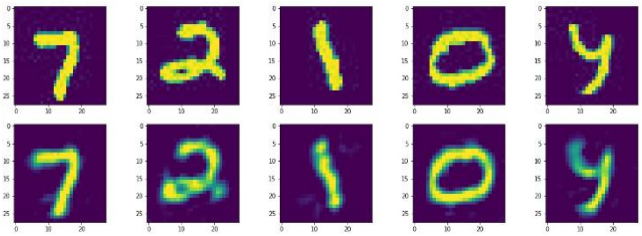

Fig 10: 1st Row Original Images; 2nd Row Compressed Images

From the Fig 9, we can conclude that autoencoder was able to compress and reconstructs the image with normal quality which is not better than NMF. The compression of ratio of auto encoder method is ~1.1.

The following table shows us the PSNR Values of Autoencoder tested on the following handwritten digits.

TABLE I : PSNR Ratio of auto encoder

Letters	PSNR Ratio
Seven	32.67
Two	31.46
One	34.08
Zero	31.94
Four	32.50

4.2.2 *Restricted Boltzmann Machines*: The proposed method of RBM have tested with different set of test images. Here are the 5 sample test images with PSNR Ratios.

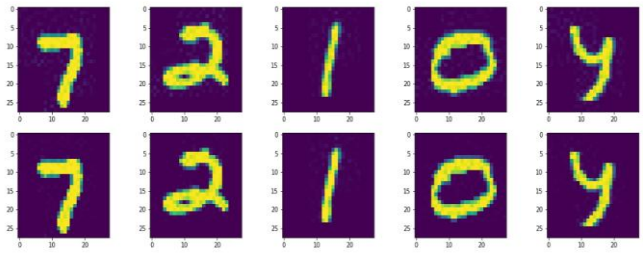

Fig 11: 1st Row Original Images; 2nd Row Compressed Images

From the Fig 11, we can conclude that RBM is able to compress and reconstruct the image with normal quality which is better than auto encoders but lesser than NMF. The compression of ratio of auto encoder method is ~1.4.

The following table shows us the PSNR Values of RBM of each and every letter.

TABLE III : PSNR Ratio of RBM

Letters	PSNR Ratio
Seven	36.66
Two	36.51
One	42.11
Zero	35.55
Four	37.20

4.2.3 Non-negative Matrix Factorization: The proposed method of NMF has tested with different set of test images. Here are the 5 sample test images with PSNR Ratios.

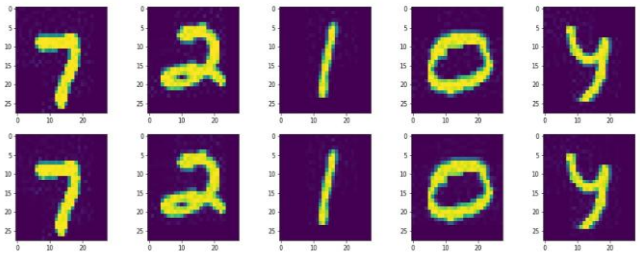

Fig 12: 1st Row Original Images; 2nd Row Compressed Images

From the Fig 12, we can conclude that NMF is able to compress and reconstructs the image with normal quality which is better than Autoencoders and also better than RBM. The compression of ratio of NMF is ~1.8.

The following table shows us the PSNR Values of NMF of each and every letter.

TABLE IIIII : PSNR Ratio of NMF

Letters	PSNR Ratio
Seven	41.88
Two	38.42
One	47.03
Zero	39.52
Four	41.04

4.2.4 Projective Non-negative Matrix Factorization: The proposed method of P-NMF has tested with different set of test images. Here are the 5 sample test images with PSNR Ratios.

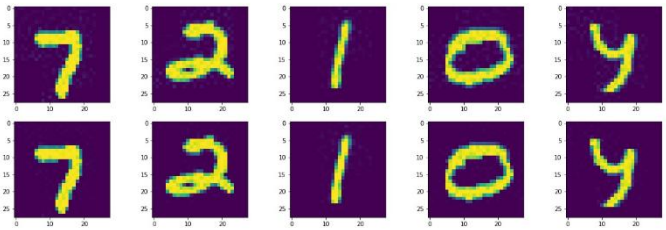

Fig 13: 1st Row Original Images; 2nd Row Compressed Images

From the Fig 13, we can conclude that PNMF is able to compress and reconstructs the image with normal quality which is better than Autoencoders and also better than RBM and not better than NMF. The compression of ratio of PNMF is ~1.8.

The following table shows us the PSNR Values of PNMF of each and every letter.

TABLE IV : PSNR Ratio of PNMF

Letters	PSNR Ratio
Seven	36.40
Two	35.17
One	39.59
Zero	34.58
Four	36.51

4.2.5 *Least Squares Non-Negative Matrix Factorization:* The proposed method of LSNMF has tested with different set of test images. Here are the 5 sample test images with PSNR Ratios.

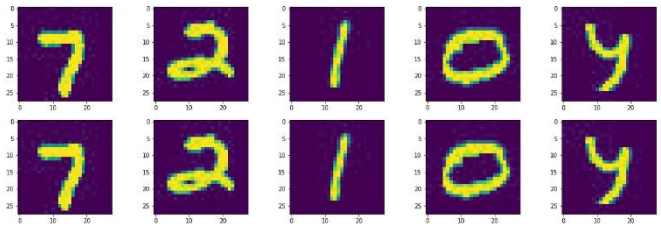

Fig 14: 1st Row Original Images; 2nd Row Compressed Images

From the fig 11 we can conclude that LSNMF is able to compress and reconstruct the image with normal quality which is better than auto encoder and better than RBM,PNMF,NMF.the compression ratio of LSNMF is 1.4

TABLE V : PSNR Ratio of LSNMF

Letters	PSNR Ratio
Seven	45.46
Two	45.93
One	48.06
Zero	48.48
Four	46.57

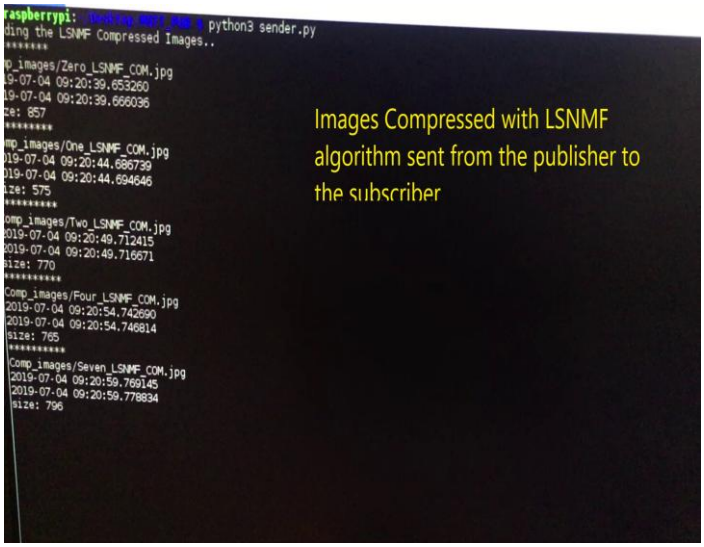

Fig 15 Images compressed with LSNMF sent from publisher to subscriber

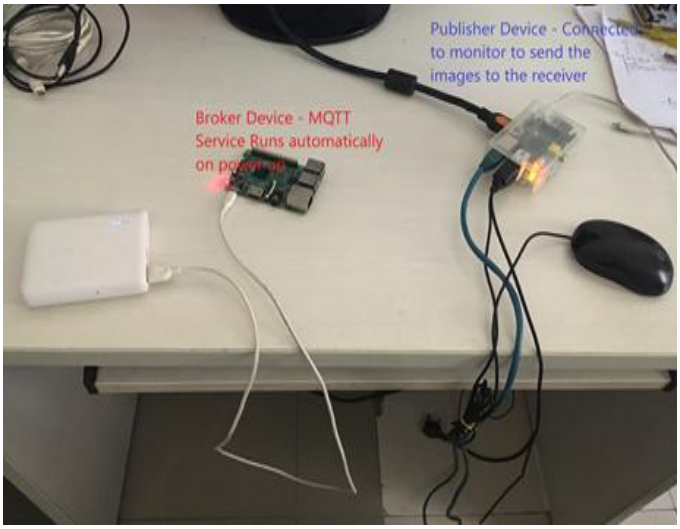

Fig 17: Images sent from subscriber device

4.3 Performance Analysis of WSN via MQTT: We have used an MQTT Transfer protocol is to transmit the original images and compressed images throughout the Wireless sensor network.

V. CONCLUSION

Image compression is a relevant research field in WSNs.It is difficult to find a comprehensive method of image compression because of the complexity in implementing sensor networks. Auto encoder, RBM, NMF, LSNMF, PNMF based image compression techniques were proposed in this paper and implemented successfully across WSN. After calculating the PSNR values of those methods, we have concluded that NMF has got a better performance when compared to Autoencoder and RBM. We have implemented and run these codes on Raspberry Pi hardware and transmitted those compressed images across the WSN. The transmission time was measured for different images for different methods implemented and it was found to be fast and reliable with excellent data integrity.

A CoAP transfer protocol can be used to enhance further our work as it can provide better security and reliable data transfer.

REFERENCES

1. Abdullah,AB., Ragab,K., & Zaman.,N.(2012). Wireless Sensor Networks and Energy Efficiency. *Published by IGI Global*, pp 1-18
2. Theis, Lucas, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. "Lossy Image Compression with compressive autoencoders." arXiv preprint arXiv:1703.00395 (2017).
3. Berry.M, Browne.M, Langville.A, Pauca.V, and Plemmons.R (2007) Algorithms and applications for approximate nonnegative matrix factorization. Computational Statistics & Data Analysis, 52:155–173,
4. Buciu.I,Nikolaidis.N, and Pitas.I, Nonnegative matrix factorization in polynomial feature space. IEEE Trans. on NN, 19(6):1090–695, 2008.
5. Cavalcanti.D, Das.S, Wang.J, and Challapali.K, "Cognitive radio based wireless sensor networks," *Proceedings of the 17th International Conference on Computer Communication and Networks*, 2008, ICCCN '08, pp. 1–6
6. Ding., C, He., X & Simon., D. H (April 2005) On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering. *Proc. SIAM Int'l Conf. Data Mining (SDM'05)*, pp. 606-610.
7. Friend.D.H, "Cognitive networks: Foundation to applications," PhD diss., Electrical and Computer Engineering, *Virginia Polytechnic and State University, Blacksburg*, Va., March 6, 2009.
8. Hoyer.P.O,Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.
9. Huang., K, Sidiropoulos., N & Swami (2014) A. Non-negative matrix factorization revisited: Uniqueness and algorithm for symmetric decomposition. *IEEE Trans. Signal Process*.pp.211–224. [CrossRef]
10. Lee.D.D and Seung.H.S, Learning the parts of objects by nonnegative matrix factorization. Nature, 401:788–791, 1999.
11. Li., T. & Ding., C.(2006) The Relationships Among Various Nonnegative Matrix Factorization
12. Methods for Clustering. *Proc. IEEE Int'l Conf. on Data Mining (ICDM'06)*, pp. 362-371.
13. Lin.C.J, Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.
14. Minoli, D., Sohraby, K., & Znati, T. (2007). Wireless sensor networks, technology, protocols, and applications. *John Wiley & Sons, Inc Hoboken, NJ*:. doi:10.1002/047011276X
15. Shenai.K and Mukhopadhyay.S, "Cognitive sensor networks," *IEEE 26th International Conference on Microelectronics (MIEL)*, May 2008, pp. 315–320
16. Wang, K.J., & Zuo, C.T.(2014) Improvements of non-negative matrix factorization for image extraction. *Appl. Res. Comput*.(In Chinese)
17. Cheng, Chunling, Shu Wang, Xingguo Chen, and Yanying Yang. "A Multilayer Improved RBM Network Based Image Compression Method in Wireless Sensor Networks." International Journal of Distributed Sensor Networks, (March 2016).
18. S. Aruna Deepthi, E. Sreenivasa Rao, M. N. Giri Prasad, "RTL Implementation of image compression techniques in WSN"IJECE"International journal of electrical and computer engineering vol-9, no-3 June 2019. http://doi.org/10.11591/ijece.v9i3.pp1750-1756
19. Song, Younghoon, Hyungsik Shin, and Jeongyeup Paek. "Lightweight Server-Assisted HK Compression for Image-Based Embedded Wireless Sensor Network." IEEE Systems Journal 99 (2018): 1-11.
20. Singh, Ramnik, and Anil Kumar Verma. "Efficient image transfer over WSN using cross layer architecture." Optik 130 (2017): 499-504.
21. Atmoko, R. A., R. Riantini, and M. K. Hasin. "IoT real time data acquisition using MQTT protocol." In Journal of Physics: Conference Series, vol. 853, no. 1, p. 012003. IOP Publishing, 2017.
22. Jaladi, Aarti Rao, Krishna Khithani Pankaja Pawar,Kiran Malvi and Gauri Sahoo.:Environmental monitoring using wireless sensor networks(WSN) based on IOT".Int.Res.J.Eng.Technol 4,no.1(2017)
23. M. Wegmuller, J. P. von der Weid, P. Oberson, and N. Gisin, "High resolution fiber distributed measurements with coherent OFDR," in *Proc. ECOC'00*, 2000, paper 11.3.4, p. 109.
24. R. E. Sorace, V. S. Reinhardt, and S. A. Vaughn, "High-speed digital-to-RF converter," U.S. Patent 5 668 842, Sept. 16, 1997.
25. (2002) The IEEE website. [Online]. Available: http://www.ieee.org/
26. M. Shell. (2002) IEEEtran homepage on CTAN. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/supported/IEEEtran/
27. Yang.J, Yan.S, Fu.Y,Li.X, and Huang.T. ,Non-negative graph embedding. *In CVPR*, pages 1–8, 2008.
28. Youssef.W and Younis.M, "A cognitive scheme for gateway protection in wireless sensor network," *Applied Intelligence Journal* 29, no. 3, 2008, pp. 216–227.
29. Zhang.D, Zhou.Z.H, and Chen.S,Non-negative matrix factorization on kernels. *In PRICAI*, pages 404–412, 2006.
30. Ali Taylan Cemgil,"Bayesian inference for Nonnegative Matrix Factorization Models",Journal of computational intelligence and Neuroscience,feb 2009.
31. Antonie Liutkus ,Derry, Roland"Cauchy nonnegative matrix factorization "IEEE workshop on applications of signal processing to Audio and Acoustics (WASPAA), Oct 2015, New York ,United states<hal -01170924>
32. A robotized wireless sensor network based on MQTT cloud computing Sevil A. Ahmed and Andon V. Topalov and Nikola Georgiev Shakev}, 2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)},pg-1-6
33. Daoqiang Zhang, Wanquan Liu"An Efficient Nonnegative Matrix Factorization Approach in Flexible Kernel Space IJCAI 2009 Proceedings of the Twenty-First International Joint Conference on Artificial intlligence,Pasadena,california,USA 2009 - aaai.org.
34. Shikang Kong 1 , Lijuan Sun 1,2,3,*, Chong Han 1,2,3 and Jian Guo 1,2,3" An Image Compression Scheme in Wireless Multimedia Sensor Networks Based on NMF" MDPI Information journal -24 February 2017

BIOGRAPHIES OF AUTHORS

S. Aruna Deepthi received B.Tech., Degree in Electronics and Communications Engineering from the Nagarjuna University Andhra Pradesh, India, and the M.E. Degree in signals and system processing from the Osmania University in India, She is currently working as Assistant Professor in Department of Electronics and communications in Vasavi college of Engineering and pursuing PhD in the Department of Electronics and communications Engineering at JNTU Anantapuramu India. Her current research interests include Image processing, WSN, energy conservation, clustering, routing algorithms, Digital Communications, Machine learning.

Dr. E. Sreenivasa Rao received his bachelor's degree from Andhra University; his master degree in computers from JNTU Hyderabad, Master's degree in Opto electronics from University of Cochin, his doctoral degree from JNTU Hyderabad. At present he is working as Professor and HoD in Vasavi college of Engineering. His research interests includes Microcontrollers, communications, Wireless Networks, computer Networks, computer organization.

Dr. M. N. Giri Prasad Worked as HOD, ECE, JNTUACE, and Pulivendula from 2006 to 2011. He worked as Principal, JNTUACE, Pulivendula from Jan 2010 to Sept. 2010.. He is presently working as Director of admissions JNTUA, Professor, Department of ECE, JNTUA Andhra Pradesh India. His research interests are in the areas of Bio-Medical Signal Processing, Image and signal Processing, Microcontrollers, and Communications, Embedded systems, Digital system design and Instrumentation.