Rate-distortion Balanced Data Compression for Wireless Sensor Networks

Mohammad Abu Alsheikh, *Student Member, IEEE,* Shaowei Lin,
Dusit Niyato, *Senior Member, IEEE,* and Hwee-Pink Tan, *Senior Member, IEEE*

***Abstract*—This paper presents a data compression algorithm with error bound guarantee for wireless sensor networks (WSNs) using compressing neural networks. The proposed algorithm minimizes data congestion and reduces energy consumption by exploring spatio-temporal correlations among data samples. The adaptive rate-distortion feature balances the compressed data size (data rate) with the required error bound guarantee (distortion level). This compression relieves the strain on energy and bandwidth resources while collecting WSN data within tolerable error margins, thereby increasing the scale of WSNs. The algorithm is evaluated using real-world datasets and compared with conventional methods for temporal and spatial data compression. The experimental validation reveals that the proposed algorithm outperforms several existing WSN data compression methods in terms of compression efficiency and signal reconstruction. Moreover, an energy analysis shows that compressing the data can reduce the energy expenditure, and hence expand the service lifespan by several folds.**

***Index Terms*—Lossy data compression, error bound guarantee, compressing neural networks, Internet of things.**

I. Introduction

By 2020, 24 billion devices are expected to be interconnected over the Internet of things (IoT) technology in which wireless sensor networks (WSNs) form an intrinsic operational component [2]. In these large-scale sensing networks, data compression is required for encoding the data collected from sensors into fewer bits, and hence reducing energy and bandwidth consumption. However, the computational burdens of the intended compression algorithms must be considered. Specifically, traditional data compression schemes from information and coding theory cannot be directly applied to a resource limited framework like WSNs as they are designed to optimize storage rather than energy consumption [3]. Data compression enhances the functionality of WSNs in three main ways. Firstly, compression at cluster heads, gateways, or even within sensor nodes is one key ingredient in prolonging network lifetime [3], [4]. Secondly, archiving the sensing raw data over several years requires a tremendous amount of storage that ranges from terabytes to petabytes [5]. Thirdly, data compression increases the networking security by sending compressed data instead of the raw one. In particular, an intruder must fully access the data decompression procedure along with its parameters to reconstruct the raw data. The security and data privacy problem is receiving more attention especially in human-centric sensing and wireless body area networks [6], [7].

Once a deep understanding of monitored phenomena is achieved, the precise absolute readings of the sensors are not required, and extending the network lifespan is favored while collecting data within tolerable error margins [8]. Lossy data compression methods in WSNs are preferable over the lossless ones as they provide better compression ratio at lower computational cost [3]. However, most traditional lossy data compression algorithms in WSNs lack an error bound guarantee mechanism due to the high computational demand of data decompression and reconstruction [3]. Moreover, the complexity of the decompression routine becomes critical when the data destination is another resource-constrained node in the network. Thus, the computational complexity of data decompression is still an important concern.

The above discussion motivates the need for a solution that collectively supports the aforementioned design essentials. Briefly, our main contributions in this paper are as follows.

1) We propose a low-cost (both compression and decompression) lossy data compression technique with error bound guarantee. The routines for compression and decompression are implemented using only linear and sigmoidal operations. The compressed data features can be fed to machine learning algorithms [9] to automatically predict human activities and environmental conditions.
2) Unlike many conventional methods, our unified method is easily customized for both temporal and spatial compression. This allows the design of a uniform sensing framework that does not require many dedicated compression solutions, i.e., one for each application.
3) The proposed compression algorithm introduces a free level of security as an offline learned decompression dictionary is needed to recover the data. Other conventional data compression algorithms, such as [8], [10]–[12], lack this benefit as they are based on static procedures and do not use encoding dictionaries.

Experiments on real world datasets show that the algorithm outperforms several well-known and traditional methods for data compression in WSNs. Furthermore, we show that the

A preliminary version of this paper has appeared in the proceedings of the 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, see [1].

M. Abu Alsheikh is with the School of Computer Engineering, Nanyang Technological University, Singapore 639798 and also with the Sense and Sense-abilities Programme, Institute for Infocomm Research, Singapore 138632 (stumyhaa@i2r.a-star.edu.sg). S. Lin is with the School of Engineering Systems and Design Pillar, Singapore University of Technology and Design, Singapore 487372 (shaowei_lin@sutd.edu.sg). D. Niyato is with the School of Computer Engineering, Nanyang Technological University, Singapore 639798 (dniyato@ntu.edu.sg). H.-P. Tan is with the School of Information Systems, Singapore Management University, Singapore 188065 (hptan@smu.edu.sg).

data compression using the proposed algorithm helps in reducing the data consumption in WSNs.

The rest of the paper is organized as follows. We first summarize related works in spatial and temporal compression of sensor data in Section II. Section III presents the problem formulation and describes some network topologies where data compression is befitting. We then provide a mathematical overview on neural network autoencoders, and propose a compression algorithm that exploits data spatial and temporal correlations using autoencoders while providing an error bound guarantee in Sections IV and V, respectively. Then, we evaluate and discuss the performance of our algorithm in experiments with actual sensor data in Section VI. Finally, Section VII concludes the paper by outlining our key results and potential future work.

II. Related Work

We identify a wide variety of coding schemes in the literature (e.g., [3], [13], [14]) and discuss some important solutions for signal compression in WSNs in the following.

A. Limitations of Conventional WSN Compression Methods

The lightweight temporal compression (LTC) algorithm [10] is a simple method to compress the environmental data. LTC is a linear method that represents a time series readings by using a set of connecting lines. A similar model-based approach is the piecewise aggregate approximation (PAA) algorithm [15] that reduces the dimensionality of source data by generating series of discrete levels. On the negative side, both LTC and PAA are less efficient when the data values change significantly over time even if the data periodically follows the same pattern and values. Moreover, they can only be used for temporal data compression as their use for spatial compression is usually inefficient.

Principal component analysis (PCA), also known as the Karhunen-Loeve transform, has been widely used to extract (linear) correlations among sensor nodes (e.g., [16]–[19]). Furthermore, a major scheme in the development of lossy data compression relies on the transformation of raw data into other data domains. Examples of these methods include discrete Fourier transform (DFT), fast Fourier transform (FFT) [14], and the different types of discrete cosine transforms (DCT) [11]. Naturally, these transformation methods exploit the knowledge of the application to choose the appropriate data domain that discards the least data content. However, such algorithms suffer from low performance when used to compress data spatially or when noises are present in the collected readings.

B. Limitations of Compressive Sensing (CS)

On the condition that a sparse representation[1] of a given signal is achievable, compressive sensing (CS) can efficiently transform the signal into a compressed form, which will be used later to recover an approximation of the original signal. In [20]–[23], the adoption of compressive sensing in WSNs is presented. Applying CS in WSNs has many limitations. Firstly, the assumption of input signal sparsity is strong and requires careful consideration in real-world deployments. Specifically, WSN data may not be sparse in the conventional data representations, such as the time, wavelet, and frequency domains. Similarly, linear sparse coding methods, such as dictionary learning [24], result in poor reconstruction fidelity due to the typical nonlinear correlations in WSN data. Secondly, introducing a few noisy readings may corrupt the sparse data representation (e.g., this issue is shown in [21] for the DCT method). Thirdly, CS requires the transmission of 3-4 times the number of nonzero values in the original signal for effective signal recovery [21]. This can result in low compression performance in real-world WSN deployments. Finally, the complexity of CS data decompression hinders the development of error bound guarantee for CS-based compression methods in WSNs.

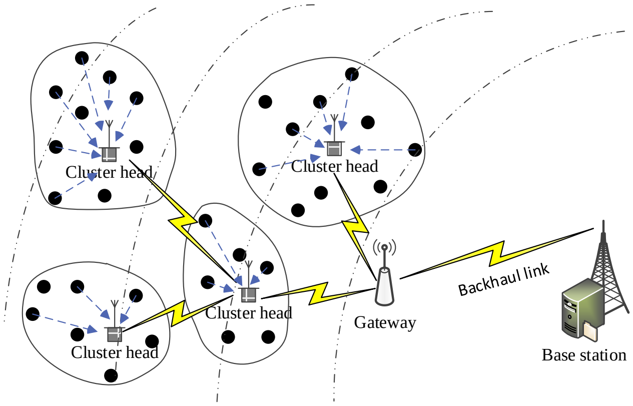


Fig. 1: System model for data aggregation and funneling application over a backhaul link. Data can be compressed at the sensors, cluster heads, or gateway.

To address these limitations of existing methods, we propose a low-complexity lossy compression algorithm that exploits the nonlinear spatial-temporal correlations in WSN data and provides an error bound guarantee. Our algorithm automatically discovers intrinsic features in WSN data for efficient data representation, rather than relying on given features which may not suit the data well.

III. System Model and Problem Formulation

Throughout the paper, we will use the following standard notational conventions: A matrix is denoted by a bold uppercase letter, a vector is represented by a bold lowercase letter, and a scalar is denoted by a lowercase letter. Finally, functions and constants are named by uppercase letters.

In this section, we give an overview of the problem considered in this paper including the data compression schemes (i.e., spatial and temporal compression). As shown in Figure 1, assume that each observed sample of sensor i $(i = 1, \dots, N)$ at time instant t $(t = 1, \dots, M)$ is formed as

$$x_i[t] = x_i^*[t] + w_i[t], \tag{1}$$

[1] A signal representation is considered sparse if it contains most or all information of the original signal using relatively small number of nonzero components.

where i is the spatial location identifier and t is the discrete time identifier. Consequently, all sensors are assumed to be synchronized according to the discrete-time model. N is the number of spatial locations which is equal to the number of sensor nodes (i.e., each location is covered by a single sensor). $x_i^*[t]$ is the noiseless physical phenomenon value (e.g., a temperature value), and the noise values $\{w_i[t] \sim \mathbb{N}(0, \sigma_w^2)\}_{i=1}^{N}$ are i.i.d random Gaussian variables with zero mean and variance σ_w^2 that depends on the hardware accuracy. Moreover, we assume that $\varphi_1 < |x_i[t]| < \varphi_2$ which is defined as the dynamic range of the sensors with φ_1 and φ_2 as constants that are usually given in hardware data sheets. Thereby, any sample value that falls outside this sensing range is considered as an outlier reading (e.g., generated by a defective sensor). For example, the RM Young wind monitoring sensor (model 05103) [25] measures the wind speed in the range of 0 to 100 m/s. Therefore, any reading beyond this range is considered as invalid data and should be eliminated.

Naturally, compression algorithms exploit the redundancy to extract spatial and temporal correlations from data. The choice of an optimal data compression scheme for a WSN is affected by network topology, routing algorithm, and data patterns [26]. To simplify the notations, we will consider the data vector denoted by $\mathbf{x} \in \mathbb{R}^L$, $L \in \{N, M\}$, that is formed from a single location's measurements over time (a temporal data vector from a single sensor) or by combining many locations' measurements at a single time instant (a spatial data vector from many locations). Assuming perfect (without any missing or outlier values) data samples in $\mathbf{x}$, the data compression (either temporal or spatial) is intended to represent $\mathbf{x}$ in a compressed form $\mathbf{y} \in \mathbb{R}^K$, where $K < L$. The compressed data is sent over a wireless channel to a base station (BS) over a backhaul link or using multihop transmissions. The BS must be able to compute a reconstruction of the original data $\hat{\mathbf{x}}$. The reconstruction may be required to be within a guaranteed threshold (i.e., a tolerable error margin). Next, we give an overview of the data compression schemes considered in this paper and the network topologies that fit each scheme.

A. Temporal (Intrasensor) Compression

This compression scheme exploits data redundancy of one sensor over time. Each sensor independently compresses its own data before transmission. Temporal compression is independent of the network topology as it does not require inter-sensor communication [26]. The temporal compression achieves maximum performance when the observed phenomenon changes slightly over time, such as hourly temperature or humidity readings. At a specific location i, the temporal data vector is formed as $\mathbf{x} = \{x_i[t]\}_{t=1}^{M} \in \mathbb{R}^M$. M is designed to fit the physical phenomenon cycle and the sampling rate.

B. Spatial (Inter-sensor) Compression

In a dense network, data collected by neighboring sensors is highly correlated [27]. Spatial compression investigates the disseminated data patterns among different sensors over the area. Therefore, the performance of the spatial compression algorithm will be affected by the network topology and sensor deployment. Unlike temporal compression that considers only a single sensor, spatial compression considers a WSN with N nodes. At a specific time instant t, the samples of all sensors are used to construct a single data vector as $\mathbf{x} = \{x_i[t]\}_{i=1}^{N} \in \mathbb{R}^N$.

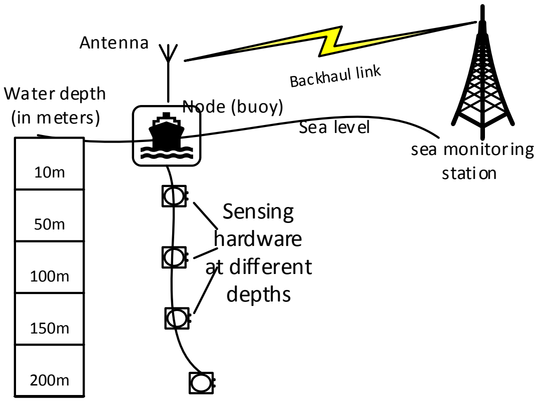


Fig. 2: A wireless sensor node to monitor the water condition (e.g., pH level) at different depths using sensing components fixed by a cable to a buoy.

Another fitting architecture for spatial compression is demonstrated in Figure 2. A buoy sensor node is used for data monitoring at different depths of the sea, where each node comprises a collection of sensors and a single transmission unit. The data is compressed at the buoy node by exploiting the spatial correlation among sensor's readings before transmitting them to a sea monitoring station.

The next section gives an overview of a special type of artificial neural networks called the autoencoder network. The discussion describes the procedure that is followed to generate a compressed data representation at a hidden layer and reconstructed data values at an output layer. Our algorithm will be later developed based on this formulation.

IV. Neural Autoencoders (AEs)

Artificial neural networks (ANNs) have been successfully used in the development of novel solutions for WSNs as they can capture nonlinear structures in data [9]. For example, an ANN-based method for minimizing environmental influences on sensor responses was proposed in [28]. It has been shown in [29] that ANNs are a solid tool for maximizing sensing coverage. This paper presents an appealing application of ANNs for data compression in WSNs. The key technical challenges of this application are (i) learning nonlinear spatio-temporal correlations of WSN data, (ii) enabling low-cost data compression and decompression, (iii) ensuring data reconstruction within tolerable error margins, and (iv) minimizing WSN energy consumption.

An autoencoder (or auto-associative neural network encoder) is a three-layer neural network that maps an input vector $\mathbf{d} \in \mathbb{R}^L$ to a hidden representation $\mathbf{y} \in \mathbb{R}^K$ and finally to an

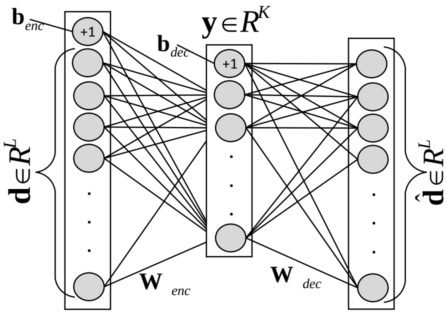


Fig. 3: Using AE to project the data to a lower dimensional representation ($K \ll N$).

output vector $\hat{\mathbf{d}} \in \mathbb{R}^L$ that approximates the input $\mathbf{d}$, as shown in Figure 3. The vectors satisfy

$$\mathbf{y} = F\left(\mathbf{W}_{enc}\mathbf{d} + \mathbf{b}_{enc}\right) \tag{2a}$$

$$\hat{\mathbf{d}}_{\boldsymbol{\theta}}(\mathbf{d}) = F\left(\mathbf{W}_{dec}\mathbf{y} + \mathbf{b}_{dec}\right) \tag{2b}$$

$$F\left(\upsilon\right) = \frac{1}{1 + \exp(-\upsilon)} \tag{2c}$$

where $\boldsymbol{\theta} := [\mathbf{W}_{enc}, \mathbf{b}_{enc}, \mathbf{W}_{dec}, \mathbf{b}_{dec}]$ are real-valued parameters that must be learned by a suitable training algorithm, and $F\left(\cdot\right)$ is the sigmoid function. Note that other nonlinear function such as the hyperbolic tangent can also be used. The parameters $\mathbf{W}_{enc}$ and $\mathbf{b}_{enc}$ are the encoding weight matrix and bias, while $\mathbf{W}_{dec}$ and $\mathbf{b}_{dec}$ are the decoding weight matrix and bias, respectively. The entries of $\mathbf{y}$ and $\hat{\mathbf{d}}$ are sometimes called activations.

To learn optimal neural weights $\boldsymbol{\theta}$ using training data $\mathbf{D}$, we define the cost function of the basic autoencoder (AE) as

$$\Gamma_{\text{AE}}\left(\boldsymbol{\theta}, \mathbf{D}\right) = \frac{1}{|\mathbf{D}|} \sum_{\mathbf{d} \in \mathbf{D}} \frac{1}{2} \left\| \mathbf{d} - \hat{\mathbf{d}}_{\boldsymbol{\theta}}(\mathbf{d}) \right\|^2. \tag{3}$$

This function penalizes the difference between each input vector $\mathbf{d}$ and its reconstruction $\hat{\mathbf{d}}_{\boldsymbol{\theta}}(\mathbf{d})$. Consequently, the optimal neural weights may be computed using standard optimization algorithms such as the L-BFGS.

Different variants of the basic AE have been introduced in the literature to discourage the neural network from overfitting the training data [30]. Generally speaking, these regularization methods penalize the neural weight characteristics or the hidden layer sparsity characteristics.

Weight decaying autoencoder (WAE): In this variant, the cost function is defined with an extra weight decay term:

$$\Gamma_{\text{WAE}}\left(\boldsymbol{\theta}, \mathbf{D}\right) = \Gamma_{\text{AE}}\left(\boldsymbol{\theta}, \mathbf{D}\right) + \frac{\alpha}{2}\left(\left\|\mathbf{W}_{enc}\right\|^2 + \left\|\mathbf{W}_{dec}\right\|^2\right) \tag{4}$$

where $\left\|\mathbf{W}\right\|^2$ represents the sum of the squares of the entries of a matrix $\mathbf{W}$, and α is a hyperparameter[2] that controls the contribution from the weight decay term.

Sparse autoencoder (SAE): This version extracts a sparse data representation at the hidden layer. In particular, we want most of the entries of $\mathbf{y}$ to be close to zero. Sparsity is encouraged by adding the Kullback–Leibler (KL) divergence function [31]:

$$\Gamma_{\text{SAE}}\left(\boldsymbol{\theta}, \mathbf{D}\right) = \Gamma_{\text{WAE}}\left(\boldsymbol{\theta}, \mathbf{D}\right) + \beta \sum_{k=1}^{K} \text{KL}(\rho || \hat{\rho_k}) \tag{5a}$$

$$\text{KL}(\rho || \hat{\rho_k}) = \rho \log_e \frac{\rho}{\hat{\rho_k}} + (1-\rho)\log_e\left(\frac{1-\rho}{1-\hat{\rho_k}}\right) \tag{5b}$$

where β is a hyperparameter that controls the sparsity weight, ρ is the sparsity parameter (target activation) that is chosen to be close to zero, and $\hat{\rho_k}$ is the average activation of the k-th node in the hidden layer.

Next, the proposed algorithm is described in more details, and a discussion is provided to signify the advantages of our AE-based compression algorithm. Moreover, a method is presented to ensure data compression within a tolerable error margin (i.e., an error bound guarantee). Finally, simple but important methods for data preparation and missing data imputation are also presented.

V. Lossy Compression with Error Bound Guarantee

We propose to apply the autoencoder to the data compression and dimensionality reduction problem in WSNs to represent the captured data using fewer bits as demonstrated in Figure 4. The algorithm enables compressed data collection with tolerable error margins, and it contains three main steps: historical data collection using the sensor nodes, offline training and modeling at the BS, and online data temporal or spatial compression. The proposed algorithm is motivated by several reasons related to WSN characteristics, as well as the ability of AEs to automatically extract features in the data.

1) AEs are used to extract a suitable, low-dimensional code representation that retains most of the information content of the original data. Besides data compression, these intrinsic features are integral for data analytics and visualization algorithms [32], e.g., classification problems.
2) Sensor networks are deployed in a variety of distinct scenarios with different network structures and data patterns. The proposed algorithm has the flexibility of supporting many scenarios using one unified technique.
3) Finally, after learning the AE's parameters, the process of data encoding and decoding are simple and can be programmed with a few lines of code.

A. Missing Data Imputation

Missing WSN data can occur due to wide variety of causes such as malfunctioning node, communication failure and interference, and unsynchronized sampling of the sensors. For missing data imputation, we use a simple naive method as shown in Figure 5. Suppose that the entry x_{ij} in the aligned matrix is missing, where i and j are the time and sensor indices. Let S be the set of observed sensors at time i, and let the mean of the observed readings of sensor j be μ_j. We estimate x_{ij} as

$$\hat{x}_{ij} = \frac{\sum_{k \in S} x_{ik}}{\sum_{k \in S} \mu_k} \mu_j. \tag{6}$$

[2] A hyperparameter is a variable that is selected a priori. This differentiates a hyperparameter from a model parameter (e.g., the encoding weight) which is adjusted during the learning process.

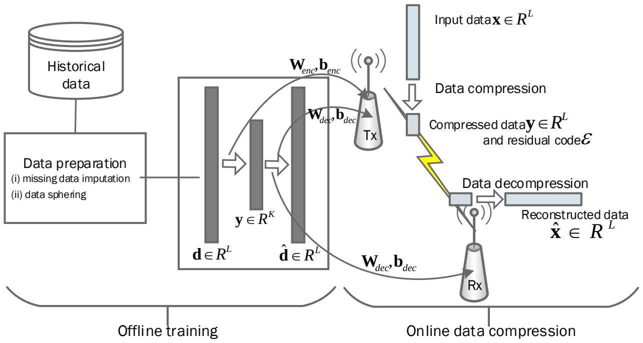

Historical data
Data preparation
(i) missing data imputation
(ii) data sphering
$\mathbf{W}_{enc}, \mathbf{b}_{enc}$
$\mathbf{W}_{dec}, \mathbf{b}_{dec}$
$\mathbf{y} \in R^K$
$\mathbf{d} \in R^L$
$\hat{\mathbf{d}} \in R^L$
Tx
Input data $\mathbf{x} \in R^L$
Data compression
Compressed data $\mathbf{y} \in R^L$ and residual code $\mathcal{E}$
Data decompression
Reconstructed data $\hat{\mathbf{x}} \in R^L$
$\mathbf{W}_{dec}, \mathbf{b}_{dec}$
Rx
Offline training
Online data compression

Fig. 4: AE adoption for data compression and dimensionality reduction in WSNs. Initially, the parameters $\mathbf{W}_{enc}, \mathbf{b}_{enc}, \mathbf{W}_{dec}$, and $\mathbf{b}_{dec}$ are adjusted during the learning stage (offline mode). Subsequently, the encoding part will be executed in the transmitter side (Tx) to achieve a compressed representation of the data. Then the receiver (Rx) will deploy the decoding part to recover a proper approximation of the original signal.

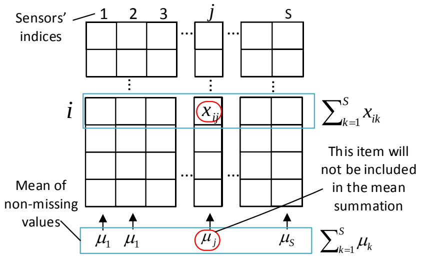


Fig. 5: Missing data prediction method.

In many sensor applications such as temperature monitoring, the naive method seems to work relatively well because of pseudo-linear correlations in the data. We chose this naive method because of the low computational resources available at the sensor nodes.

B. Data Sphering

The entries of the output vector $\hat{\mathbf{d}}$ of the AE are from the sigmoid function, so they are all between 0 and 1. Because the AE attempts to reconstruct the input vector $\mathbf{x} \in \mathbb{R}^L$, we need to normalize our input data so that the entries are also between 0 and 1. Moreover, for the AE to work, the input data vectors must be distributed somewhat uniformly near the unit sphere in $\mathbb{R}^L$. This process is called data sphering [31]. One simple method involves truncating readings that lie outside three standard deviations from the vector mean, and rescaling the remaining readings so that they are between 0.1 and 0.9. In particular, the formula is

$$\begin{aligned}\mathbf{d} &= \text{normalize}(\mathbf{x}, \sigma) \\ &= 0.5 + \frac{0.4}{3\sigma} \max\left(\min\left(\mathbf{x} - \text{mean}(\mathbf{x}), 3\sigma\right), -3\sigma\right)\end{aligned} \tag{7}$$

where $\mathbf{x}$ is the source data vector and σ is the standard deviation of the entries of $\mathbf{x}-\text{mean}(\mathbf{x})$ over all $\mathbf{x}$ in the training dataset. $\mathbf{d}$ is the data vector that is fed to the AE network.

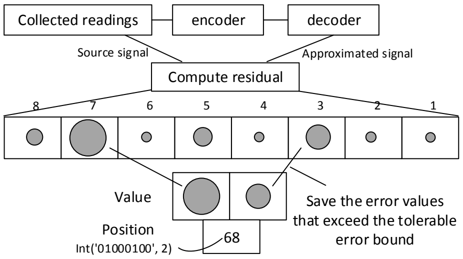


Fig. 6: The error bound mechanism performed by the transmitting node.

Furthermore, assuming the data is normally distributed, the probability that a reading is located within three standard deviations from the mean is 99.7% [33]. Conversely, given the mean m, the original data vector $\mathbf{x}$ may be reconstructed (up to truncated outliers) as

$$\mathbf{p} = \text{denormalize}(\hat{\mathbf{d}}, m, \sigma) = \frac{3\sigma}{0.4}(\hat{\mathbf{d}} - 0.5) + m, \tag{8}$$

where $\mathbf{p}$ is a reconstruction of the source data $\mathbf{x}$ by using the AE output vector $\hat{\mathbf{d}}$.

C. Error Bound Mechanism

The error bound ϵ is defined to be the maximum acceptable difference between each collected reading by the sensor and the recovered one by the receiver after receiving the compressed representation. Basically, the error bound is tuned by considering several factors such as the application requirements and the used sensors' precision. For example, the RM Young wind monitoring sensor (model 05103) [25] measures the wind speed and direction with accuracy of 0.3 m/s and 3°, respectively. Thus, setting the error bound to be equal to the sensor accuracy may be an acceptable design basis.

Let $\mathbf{p}$ be a reconstruction of $\mathbf{x}$ that is not guaranteed to be within any tolerable error margin. The error bound mechanism first computes the residual $\mathbf{r} = \mathbf{x} - \mathbf{p}$ as shown in Figure 6. Any entry of the residual vector exceeding the bound ϵ will be transmitted, using the residual code

$$\varepsilon = \text{residualCode}(\mathbf{r}, \epsilon) = \left(\mathbb{1}_J, (r_j)_{j \in J}\right) \tag{9}$$

where $J \subset \{1, \ldots, L\}$ is the set of indices j where $r_j > \epsilon$ and $\mathbb{1}_J$ is the indicator vector for the subset J (i.e. $(\mathbb{1}_J)_j = 1$ if $j \in J$ and $(\mathbb{1}_J)_j = 0$ if $j \notin J$). Conversely, given the code ε that contains error residual values, it is easy to compute an estimate of the original residual by constructing a vector whose zeros are determined by $\mathbb{1}_J$ and whose nonzero entries are given by $(r_j)_{j \in J}$. We denote this vector as residual(ε).

D. Training, Compression and Decompression

After describing different components of our algorithms, we are now ready to integrate them. We assume that all the data mentioned in this section has been aligned and that all missing values have been imputed as described in Section V-A. For the training data $\mathbf{D}$, we also ensure that outliers were removed and that readings were normalized. Let σ denote the standard deviation used in the normalization of the data.

We first learn optimal weights $\boldsymbol{\theta}$ for the autoencoder by minimizing the cost function $\Gamma_{\text{WAE}}(\boldsymbol{\theta}, \mathbf{D})$ using the L-BFGS algorithm. This computationally-intensive process only occurs once at the start of our network deployment, and the parameters $\boldsymbol{\theta}, \sigma$ are distributed to the transmitters and receivers.

The algorithms for compressing and decompressing the sensor readings are outlined in Algorithms 1 and 2, respectively. For our experiments, we send the compressed signal $(\mathbf{y}, \boldsymbol{\varepsilon}, m)$ using floating point representation for the real numbers and binary string for the indicator vector $\mathbb{1}_J$ in $\boldsymbol{\varepsilon}$. Note that all the steps have low computational complexity. Here, we also see why decoder complexity in algorithms (e.g., compressed sensing) impedes the provision of error bound guarantee because it is computationally expensive to compute the residual $\boldsymbol{\varepsilon}$. Clearly, an intruder who can receive the compressed data cannot retrieve the raw data without knowing the decoding weight matrix $\mathbf{W}_{dec}$ and bias vector $\mathbf{b}_{dec}$. This adds a free level of security to data aggregation in WSNs.

```
Algorithm 1: The online data compression
  Input: readings $\mathbf{x}$; parameters $\sigma, \mathbf{W}_{enc}, \mathbf{b}_{enc}, \mathbf{W}_{dec}, \mathbf{b}_{dec}$
  Output: signal $\mathbf{y}, \boldsymbol{\varepsilon}, m$
  begin
      $m \leftarrow$ mean($\mathbf{x}$)
      $\mathbf{d} \leftarrow$ normalize($\mathbf{x}, \sigma$)
      $\mathbf{y} \leftarrow F(\mathbf{W}_{enc}\mathbf{d} + \mathbf{b}_{enc})$
      $\hat{\mathbf{d}} \leftarrow F(\mathbf{W}_{dec}\mathbf{y} + \mathbf{b}_{dec})$
      $\mathbf{p} \leftarrow$ denormalize($\hat{\mathbf{d}}, m, \sigma$)
      $\boldsymbol{\varepsilon} \leftarrow$ residualCode($\mathbf{x} - \mathbf{p}, \epsilon$)
```

```
Algorithm 2: The online data decompression
  Input: signal $\mathbf{y}, \boldsymbol{\varepsilon}, m$; parameters $\sigma, \mathbf{W}_{dec}, \mathbf{b}_{dec}$
  Output: reconstruction $\hat{\mathbf{x}}$
  begin
      $\hat{\mathbf{d}} \leftarrow F(\mathbf{W}_{dec}\mathbf{y} + \mathbf{b}_{dec})$
      $\mathbf{p} \leftarrow$ denormalize($\hat{\mathbf{d}}, m, \sigma$)
      $\mathbf{r} \leftarrow$ residual($\boldsymbol{\varepsilon}$)
      $\hat{\mathbf{x}} \leftarrow \mathbf{p} + \mathbf{r}$
```

E. Time Complexity

Our algorithm training is computationally expensive and should be run on a server. However, the data compression and decompression, as highlighted in Algorithms 1 and 2, are lightweight. Both data compression and decompression has a linear time complexity of $\mathcal{O}(L \times K)$, where L is the input data size, and K is the compressed data size. This low computational complexity results in significant energy conservation as shown in Section VI-D.

The next section presents simulation results to show the compression performance and energy conservation of the proposed algorithm.

VI. Experimental Results

We evaluate the performance of the proposed algorithm using data from actual sensor test beds. Our datasets are divided into 10 random folds for training and testing (i.e., the cross-validation method [34] with 10 folds). In each cross-validation step, the system is trained using 9 folds and tested using the last fold. Our implementation adopts the limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [35] to tune the AE's weights during the learning stage.

A. Datasets and Performance Metrics

We evaluate our solution using the following meteorological datasets:

- Grand-St-Bernard deployment [36]: We use data from 23 sensors that collect surface temperature readings between Switzerland and Italy at an elevation of 2.3km. This dataset contains readings ranging from -32°C to 48°C, though observations suggest that the maximum and minimum values are most likely from malfunctioning sensor nodes. After removing outliers, the dataset still contains many interesting nontrivial features.
- LUCE deployment [36]: After data preparation, the used dataset contains relative humidity measurements from 90 sensors, each with around 160k readings. This dataset is an example of high resolution spatial-temporal data that is collected by WSNs to monitor an area with widely varying data characteristics.

To measure the extent that the data is being compressed, we use the following metrics:

- *Compression ratio* (CR): This metric calculates the reduction in transmitted data size due to compression which is defined as follows:
$$\text{CR}(\mathbf{x}, \hat{\mathbf{x}}) = \left(\frac{B(\hat{\mathbf{x}})}{B(\mathbf{x})}\right) \times 100 \tag{10}$$
where $B(\hat{\mathbf{x}})$ and $B(\mathbf{x})$ are the numbers of bits used to represent the transmitted and the original data, respectively.
- *Root mean squared error* (RMSE): RMSE measures the loss of data precision due to compression algorithms, i.e., compression error. An RMSE of 0 means that WSN data can be fully reconstructed without error. RMSE is defined as follows:
$$\text{RMSE}(\mathbf{x}, \hat{\mathbf{x}}) = \sqrt{\frac{1}{L}\sum_{i=1}^{L}(x_i - \hat{x}_i)^2}. \tag{11}$$
- *Coefficient of determination* (usually denoted by R^2 in statistics): This defines the proportion of variance of the original data that is reconstructed from the compressed data. An R^2 of 0.4 means that 40% of $\mathbf{x}$ is reconstructed in $\hat{\mathbf{x}}$. This metric is calculated as follows:
$$R^2(\mathbf{x}, \hat{\mathbf{x}}) = 1.0 - \frac{\sum_{i=1}^{L}(x_i - \hat{x}_i)^2}{\sum_{i=1}^{L}(x_i - \bar{x})^2}, \tag{12}$$
where $\bar{x} = \frac{1}{L}\sum_{i=1}^{L} x_i$. The data is fully reconstructed if R^2 is equal to 1.0.

The CR value determines the compression efficiency, while the RMSE and R^2 values define the reconstruction fidelity.

B. Test Example

Figure 7 provides an example of the quantities computed in the proposed compression solution. The network is trained using $28k$ records of historical dataset from the Grand-St-Bernard dataset. Figure 7a gives an example of data compression, transmission and recovery process using the AE's network. The input signal (I) is collected from the network's nodes, such that each node contributes one reading every two minutes. This input signal excites the network and a compressed signal (II) is generated and transmitted to the receiver(s) using any general routing protocol. The output signal (III) is recovered at the receiver that represents an efficient approximation of the input signal. Figure 7b shows a Hinton diagram of the learned encoding weight. The size of the squares represents the filter's magnitudes and the color represent the sign (white for negative and black for positive values). Each column shows the receptive field of each node in the other nodes. The node's receptive fields are automatically extracted to represent the spatial correlation among neighbor nodes. Figure 7c shows the RMSE over learning iterations for the training and testing datasets. The training RMSE is very high at the initial iterations but decreases with learning iterations.

C. Baselines

In this section, a simulation study of the data compression is given. This includes two main validation scenarios. Firstly, the algorithm performance without error bound guarantee is tested under spatial compression scenario using the Grand-St-Bernard deployment. This scenario is designed to test the compression ratio and reconstruction error of the proposed method against a set of conventional methods that do not provide any error bound guarantee. Secondly, a temporal compression scenario is formulated with an error bound guarantee using the LUCE deployment data. This temporal scenario tests the proposed method against the well-known LTC method [10] which provides an error bound guarantee.

1) AE Models: As shown in Figure 8, using the basic AE provides the best performance over the other AE's variants. Even though WAE and SAE are useful for classification-related tasks to avoid overfitting, we find that they degrade the AE's reconstruction performance (i.e., RMSE). This is justified as for the compression problem, the hidden layer size is less than the input size, and hence the model is less affected by the overfitting problem. However, in feature extraction and classification problems, the hidden layer could be larger than the input size and the overfitting effects are more apparent. In these cases, regulations using WAE and SAE become more important. To tune the AE's hyper-parameters, the authors of [37] describe the prohibitive task of running the cross-validation procedure using several choice of parameters. To automate this process, we employed the common strategy in machine learning by using the grid search method for model selection. Initially, this starts by specifying a list of reasonable values of each hyperparameter. Then, the algorithm is evaluated over the elements of the cross product set of the lists. In summary, the parameters that achieve the best performance on the cross-validation estimator will be chosen for real time deployment. It is important to note that the grid search method becomes ineffective for a large number of hyperparameters as the lists' product increases dramatically [38]. However, we only have two hyperparameters in the sparse AE case.

2) Spatial Compression: Without the error bound guarantee, we use the Grand-St-Bernard dataset to test the spatial compression capabilities of the proposed algorithm. The 23 sensors are assumed to be synchronized to transmit their data samples to a gateway. The gateway will spatially compress the data before sending it to the BS over a backhaul link. This data compression is a challenging task due to the non-uniform data distribution through different sensor nodes.

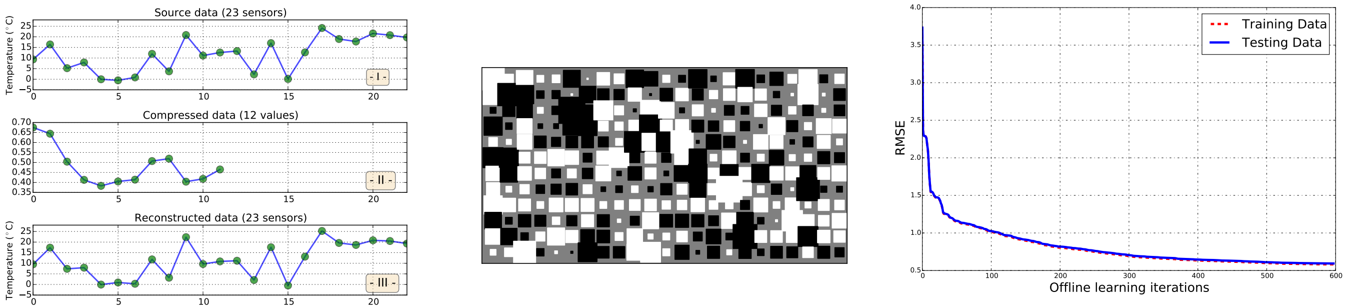
Figures 9a and 9b show that the proposed method outperforms other conventional WSN data compression methods such as PCA, DCT, FFT, and CS. These conventional methods are the main basis for most existing methods for WSN data compression [3], [14]. Our implementation of these conventional methods is based on the scikit-learn library [39]. CS samples data at a low rate than Shannon Nyquist sampling rate. Specifically, an input signal $\mathbf{x} \in \mathbb{R}^L$, $L \in \{N, M\}$ is represented as $\mathbf{x} = \mathbf{\Psi s}$, where $\mathbf{s}$ is the sparse representation of the signal with α nonzero values (called α-sparse), and $\mathbf{\Psi} \in \mathbb{R}^{L \times L}$ is the basis dictionary. We have used online dictionary learning [24] to find $\mathbf{\Psi}$. Other limitations of CS as a WSN data compression method have been discussed in Section II-B.

Based on Figure 10, we observe an important result. The average RMSE value can be misleading as WSN data compression methods without error bound guarantee can produce poor reconstruction fidelity at some time instants. Most traditional lossy data compression algorithms in WSNs lack an error bound guarantee mechanism due to the high computational demand of data decompression and reconstruction [3]. Our proposed method overcomes this limitation by using the error bound mechanism proposed in Section V-C. The proposed method with error bound of $\epsilon = 1.0$ gives a good compression ratio of CR $= 60.6\%$.

3) Temporal Compression: In the following, we compare the proposed method with the LTC method in a temporal compression scenario with error bound guarantee. We choose the LTC algorithm for bench-marking which as (1) LTC is one of the rare WSN data compression methods with error bound guarantee, and (2) several comparative studies (e.g., [14]) discussed the efficiency of the LTC algorithm over other methods in temporal compression.

Using the LUCE deployment, the temporal compression scenario is formulated such that each sensor compresses its data locally before sending it using multihop transmissions to the BS. Each sensor is assumed to sample at a rate of 1 sample every 2 minutes. Therefore, 720 samples are collected each day and sent as one compressed chunk.

Figure 11 provides the analysis of the data compression with an error bound constraint. This shows that the proposed algorithm outperforms the LTC in both RMSE (Figure 11a)



(a) (b) (c)

Fig. 7: An illustrative data compression example of surface temperature readings in a WSN containing 23 sensors: (a) an example of a data vector compression and recovery, (b) a Hinton diagram to illustrate the learned weights of the encoder, and (c) the offline learning curve over training and testing datasets.

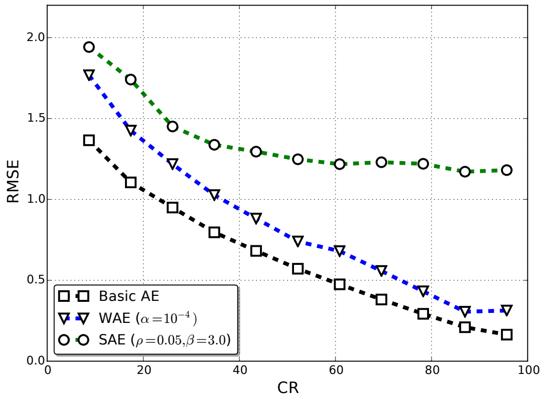


Fig. 8: Compression error (RMSE) achieved by several AE models under varied compression ratio (CR).

and compression ratio (Figure 11b). Even though the high resolution dataset of the LUCE deployment is very suitable for the LTC method as the data changes slowly between subsequent samples, the compression efficiency of the proposed algorithm is still superior. We note that LTC performs as good as AE for large error bounds, but is unable to keep the same efficiency when the error bound is small.

D. Energy Conservation by Data Compression

In this section, we consider the energy conservation of the proposed method. Traditional data compression schemes from information and coding theory cannot be directly applied to a resource limited framework like WSNs as they are designed to optimize storage rather than energy consumption [3]. Therefore, special attention should be provided to the computational burdens of any compression algorithm designed for WSNs. Otherwise, the energy consumed during CPU operations of complex algorithms might exceed the energy consumed due to sending less data over the RF module.

Again, suppose that the length of the original data vector is L and the length of the compressed data representation is K. We adapt the complexity analysis used in [14] while considering the power consumption for receiving the data which is extremely important in multihop data aggregation.

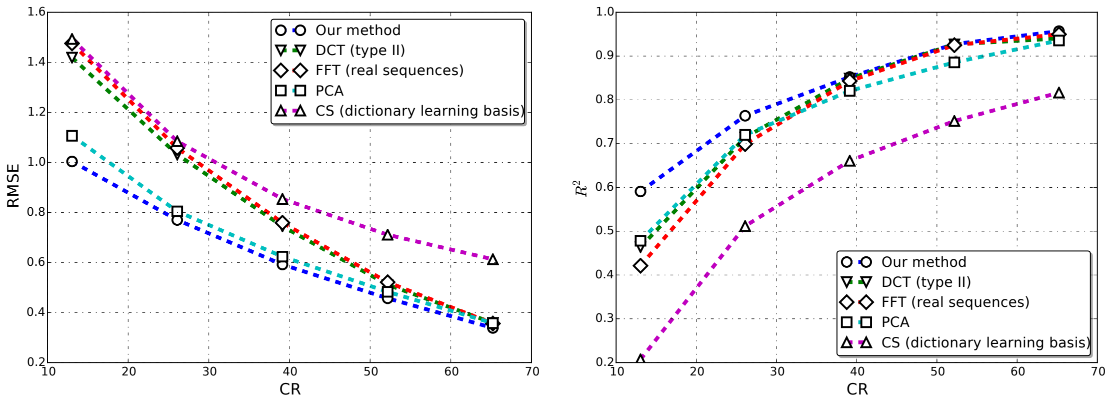
- We consider the mixed-signal MSP430 micro-controller [40] that is powered with a 16-bit CPU developed to support resource limited systems. The supply voltage is $V_{CC} = 3.3V$, the clock rate is $F_{CLK} = 3.3MHz$, and the current consumption of the complete MSP430 system during the active mode is $I_{MSP430} = 1.85mA$ (see Section 5.3.4 of [40]). Hence, the power consumption of the MSP430 micro-controller per clock cycle is

$$E_{CLK} = \frac{V_{CC} \times I_{MSP430}}{F_{CLK}} = 1.85nJ.$$

  The exponential function can be calculated using two hyperbolic functions as $\exp(v) = sinh(v) + cosh(v)$, which requires (without hardware multiplier) 52000 CPU cycles to achieve more than 6 digits of precision. This derivation is based on the Taylor series expansion. Therefore, it is important to select the number of Taylor iterations of the exponential function calculation to satisfy the precision requirements of the application [41]. The CPU cycle specifications of the basic operations are given in Table I.
- For the transmission unit, we consider the 9XTend RF module [42] that operates in the $902 - 928MHz$ frequency band with an effective data rate of $R_{XTend} = 9,600bps$ and a spread technology of frequency-hopping spread spectrum (FHSS). This module's transmission range is of up to $0.9km$ in urban areas, and up to $22km$ for ideal outdoor line-of-sight transmissions. These transmission ranges make XTend module suitable for data transmission over a backhaul link. The current consumption during the data transmission and reception are $I_{TX} = 600mA$ and $I_{RX} = 80mA$, respectively[3]. The current flow during the idle mode is near $1mA$, and hence it is ignored in our analysis. The supply voltage is set at $V_{CC} = 3.3V$. Then, the consumed energy for transmitting and receiving one bit of data is

$$S_{bit} = \frac{V_{CC} \times I_{TX} + V_{CC} \times I_{RX}}{R_{CC2420}} = 233.75\mu J.$$

[3] Some studies ignore the power consumption of data reception. However, this metric is important in multihop transmission which is performed by regular sensor nodes with limited energy budget.



(a) (b)

Fig. 9: Reconstruction fidelity of a spatial compression scenario without error bound guarantee. This shows the performance of the proposed method and conventional methods on the Grand-St-Bernard dataset. (a) Compression error (RMSE) under different values of compression ratio (CR). (b) Coefficient of determination (R^2) under different values of compression ratio (CR).

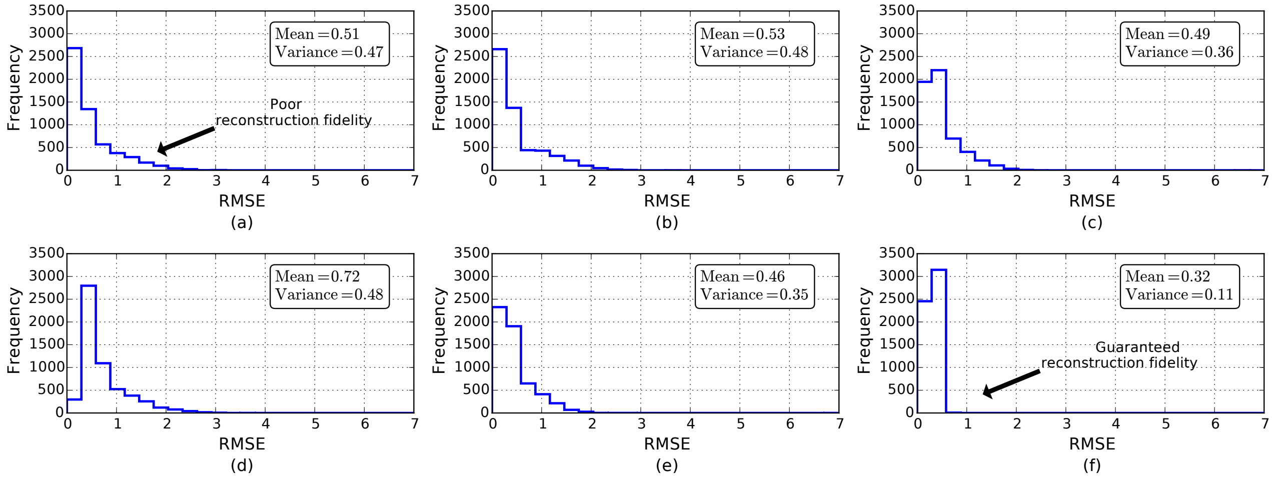


Fig. 10: Error bars of a spatial compression scenario on the Grand-St-Bernard dataset. (a)-(e) WSN data compression methods without error bound guarantee achieving a compression ratio of CR $= 52.2\%$. (a) DCT, (b) FFT, (c) PCA, (d) CS with dictionary learning basis, and (e) the proposed method. (f) The proposed method with error bound guarantee (error bound $\epsilon = 1.0$) resulting in a compression ratio of CR $= 60.6\%$.

Operation	# of CPU Cycles (FLOAT)
Addition	184
Subtraction	177
Multiplication	395
Division	405
Comparison	37
exp $(\cdot)$	52000

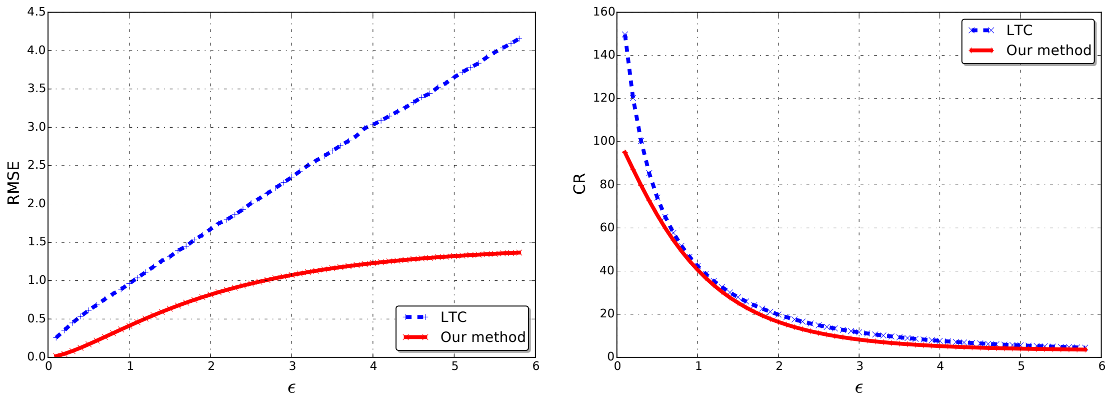
TABLE I: CPU clock cycles for the mixed-signal MSP430 micro-controller [40].

Therefore, the energy consumed by the network to transmit one bit and receive it at the next hop (one hop transmission) over the transceiver unit is approximately equal to the energy used by the micro-controller in $125,945$ CPU clock cycles[4]. Using these design components, we formulate the computational complexity (in number of clock cycles) for compressing the input vector using the AE network $C_{AE}(L, K)$ as:

$$C_{AE}(L,K) = \underbrace{(184 + 405 + 177 + 2\times 37)L}_{\text{Data normalization}} + \underbrace{(395L + 2\times 184L)K}_{\mathbf{W}_{enc}\mathbf{x}+\mathbf{b}_{enc}} + \underbrace{(184 + 405 + 52000)K}_{\text{The sigmoid function}}. \tag{13}$$

Finally, using (13), we find that the energy consumed to transmit the data with compression can be formulated as:

[4] Larger ratios of the transmission-CPU energy consumptions are even given in other studies, see [12] as an example, which is based on the hardware set and the CPU energy saving modes. These larger ratios result in more energy savings when using data compression algorithms.



(a)

(b)

Fig. 11: Analyzing a temporal compression scenario with error bound guarantee using the LUCE deployment dataset. The AE's input vector size is 720 and the hidden layer size is 20, so the achievable compression ratio is at most 97.22%. (a) Compression error (RMSE) under different values of error bound (ϵ), and (b) compression ratio (CR) under different values of error bound (ϵ).

$$E_{AE}\left(L,K\right) = \underbrace{E_{CLK}\times C_{AE}\left(L,K\right)}_{\text{CPU cost}} + \underbrace{32\text{bits}\times K\times S_{bit}}_{\text{Transmission cost}}. \tag{14}$$

The first term refers to the energy consumed to compress the data (i.e. during the CPU computations), and the second term considers the energy consumed at the transmission unit to send the compressed bits. Note that we consider a 32 bit float representation of the sensor readings. Clearly, to achieve energy savings, the energy consumption using data compression scheme must be significantly less than that of the transmission of the raw data, more formally:

$$E_{AE}\left(L,K\right) \ll 32bits \times L \times S_{bit}. \tag{15}$$

These results are illustrated in Figure 12 under different compression ratios and multihop transmissions. Specifically, Figure 12a shows the energy conservation by data compression at different compression ratios. Figure 12b shows the increased energy conservation by data compression for multihop data transmission where the forwarding nodes are typical sensor nodes with energy-limited budgets. For example, a CR of 35.56% in 5-multihop transmissions reduces the energy consumption by 2.8 folds as compared to raw data transmission. A similar result can be drawn for reliable networks in which several copies of the same packet is transmitted to ensure a packet delivery ratio.

VII. Conclusion

Instead of using computationally expensive transformations on raw data or introducing strong assumptions on data statistical models, we have proposed an adaptive data compression with feature extraction technique using AEs. Our solution exploits spatio-temporal correlations in the training data to generate a low dimensional representation of the raw data, thus significantly prolonging the lifespan of data aggregation and funneling systems. Moreover, the algorithm can optionally be adjusted to support error bound guarantee.

Recent sensor networks often monitor a variety of modalities such as temperature, humidity and illuminance. However, designing a compression algorithm for multimodal data is much more challenging than the single modal situation [13]. To study fundamental issues and design tradeoffs, we ignore the case of multimodal data in this paper and keep it for a future work. We will also study the use of sparse overcomplete representations for data compression in WSNs (i.e. when the hidden layer size is larger than the input size). Last but not least, we will explore how to integrate the presence of missing values into our autoencoder, rather than using a naive method for missing data imputation.

Acknowledgment

We thank Puay Kai Poh for valuable discussions in the early stages of the study.

References

[1] M. Abu Alsheikh, P. K. Poh, S. Lin, H.-P. Tan, and D. Niyato, "Efficient data compression with error bound guarantee in wireless sensor networks," in *Proceedings of the 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. ACM, 2014, pp. 307–311.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

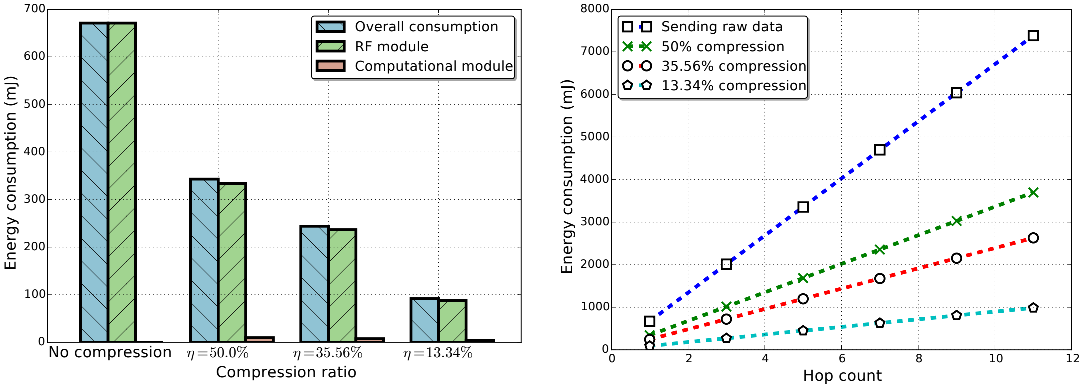
[3] M. Razzaque, C. Bleakley, and S. Dobson, "Compression in wireless sensor networks: A survey and comparative evaluation," *ACM Transactions on Sensor Networks*, vol. 10, no. 1, p. 5, 2013.

[4] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: A survey," *IEEE Wireless Communications*, vol. 14, no. 2, pp. 70–87, 2007.

[5] S. Gandhi, S. Nath, S. Suri, and J. Liu, "GAMPS: Compressing multi sensor data by grouping and amplitude scaling," in *Proceedings of the International Conference on Management of data*. ACM, 2009, pp. 771–784.

[6] M. Li, W. Lou, and K. Ren, "Data security and privacy in wireless body area networks," *IEEE Wireless Communications*, vol. 17, no. 1, pp. 51–58, 2010.

[7] M. Al Ameen, J. Liu, and K. Kwak, "Security and privacy issues in wireless sensor networks for healthcare applications," *Journal of medical systems*, vol. 36, no. 1, pp. 93–101, 2012.



(a)

(b)

Fig. 12: Energy conservation by data compression, assuming raw data of 90 sensors that is spatially compressed. (a) Energy consumption at different compression ratios. (b) Energy consumption on multihop transmissions with different hop counts.

[8] Y. Liang, "Efficient temporal compression in wireless sensor networks," in *Proceedings of the 36th IEEE Conference on Local Computer Networks*. IEEE, 2011, pp. 466–474.

[9] M. Abu Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1996–2018, 2014.

[10] T. Schoellhammer, E. Osterweil, B. Greenstein, M. Wimbrow, and D. Estrin, "Lightweight temporal compression of microclimate datasets," pp. 516–524, 2004.

[11] G. Quer, R. Masiero, D. Munaretto, M. Rossi, J. Widmer, and M. Zorzi, "On the interplay between routing and signal representation for compressive sensing in wireless sensor networks," in *Proceedings of the IEEE International Conference on Information Theory and Applications*. IEEE, 2009, pp. 206–215.

[12] C. M. Sadler and M. Martonosi, "Data compression algorithms for energy-constrained devices in delay tolerant networks," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*. ACM, 2006, pp. 265–278.

[13] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, and K. Araki, "Practical data compression in wireless sensor networks: A survey," *Journal of Network and Computer Applications*, vol. 35, no. 1, pp. 37–59, 2012.

[14] D. Zordan, B. Martinez, I. Vilajosana, and M. Rossi, "To compress or not to compress: Processing vs transmission tradeoffs for energy constrained sensor networking," *Technical Report. Department of Information Engineering, University of Padova, Padova, Italy*, 2012.

[15] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Locally adaptive dimensionality reduction for indexing large time series databases," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, vol. 30, no. 2, pp. 151–162, 2001.

[16] M. Gastpar, P. L. Dragotti, and M. Vetterli, "The distributed Karhunen–Loeve transform," *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5177–5196, 2006.

[17] G. Shen, S. K. Narang, and A. Ortega, "Adaptive distributed transforms for irregularly sampled wireless sensor networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2009, pp. 2225–2228.

[18] A. Rooshenas, H. R. Rabiee, A. Movaghar, and M. Y. Naderi, "Reducing the data transmission in wireless sensor networks using the principal component analysis," in *Proceedings of the 6th International Conference on Intelligent Sensors, Sensor Networks and Information Processing*. IEEE, 2010, pp. 133–138.

[19] H. Malik, A. S. Malik, and C. K. Roy, "A methodology to optimize query in wireless sensor networks using historical data," *Journal of Ambient Intelligence and Humanized Computing*, vol. 2, no. 3, pp. 227–238, 2011.

[20] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak, "Compressive wireless sensing," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*. ACM, 2006, pp. 134–142.

[21] C. Luo, F. Wu, J. Sun, and C. W. Chen, "Compressive data gathering for large-scale wireless sensor networks," in *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking*. ACM, 2009, pp. 145–156.

[22] L. Xiang, J. Luo, and A. Vasilakos, "Compressed data aggregation for energy efficient wireless sensor networks," in *Proceedings of the 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 2011, pp. 46–54.

[23] G. Quer, R. Masiero, G. Pillonetto, M. Rossi, and M. Zorzi, "Sensing, compression, and recovery for WSNs: Sparse signal modeling and monitoring framework," 2012.

[24] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 689–696.

[25] RM Young Company, "Wind monitor 05103 manual," 2005, http://www.youngusa.com.

[26] M. F. Duarte, G. Shen, A. Ortega, and R. G. Baraniuk, "Signal compression in wireless sensor networks," *Philosophical Transactions of the Royal Society Series A*, vol. 370, no. 1958, pp. 118–135, 2012.

[27] S. S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense microsensor network," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 51–60, 2002.

[28] J. C. Patra, P. K. Meher, and G. Chakraborty, "Development of Laguerre neural-network-based intelligent sensors for wireless sensor networks," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 3, pp. 725–734, 2011.

[29] M. Abu Alsheikh, S. Lin, H.-P. Tan, and D. Niyato, "Area coverage under low sensor density," in *Proceedings of the 11th IEEE International Conference on Sensing, Communication, and Networking*. IEEE, 2014, pp. 173–175.

[30] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 437–478.

[31] A. Ng, "Sparse autoencoder," *CS294A Lecture notes*, p. 72, 2011.

[32] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[33] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "LoOP: Local outlier probabilities," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 1649–1652.

[34] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the International Joint Conference on Artificial Intelligence*, vol. 14, no. 2, 1995, pp. 1137–1145.

[35] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[36] "Sensorscope: Sensor networks for environmental monitoring," http://lcav.epfl.ch/sensorscope-en.

[37] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2011, pp. 215–223.

[38] C. B. Do, C.-S. Foo, and A. Y. Ng, "Efficient multiple hyperparameter learning for log-linear models," in *Proceedings of the 17th Annual Conference on Advances in Neural Information Processing Systems*, 2007.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[40] L. Bierl, "MSP430 family mixed-signal microcontroller application reports," 2000, http://www.ti.com.

[41] M. Braverman and S. Cook, "Computing over the reals: Foundations for scientific computing," *Notices of the AMS*, vol. 53, no. 3, pp. 318–329, 2006.

[42] Digi International Inc., "9XTend OEM RF module: Product manual v.2.x6x," 2008, http://www.digi.com.

Mohammad Abu Alsheikh (S'14) received his B.Eng. in Computer Systems Engineering from Birzeit University, Palestine in 2011. Between 2010 and 2012, he was a software engineer working on developing robust web services, Ajax-based web components, and smartphone applications. He is currently a Ph.D. candidate in the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include machine learning in big data analytics, mobile sensing technologies, and sensor-based activity recognition.

Shaowei Lin received his Ph.D. in Mathematics under Bernd Sturmfels in 2011 from the University of California, Berkeley, where he analyzed singularities in statistical models over large data sets through the lens of modern algebraic geometry. This work was continued at Stanford University in a one-year DARPA postdoctoral collaboration with Andrew Ng's lab to explore mathematical challenges in deep learning. In 2012, he returned to Singapore to join the Institute for Infocomm Research (A*STAR) where he started the Sense-making Group in the Sense and Sense-abilities (S&S) programme. The group focused on exploiting machine learning techniques in sensor networks to create resource-efficient algorithms that exhibit higher-order intelligence. Before joining Singapore University of Technology and Design (SUTD), he oversaw deep science activities in S&S as the Deputy Head for Research.

Dusit Niyato (M'09–SM'15) is currently an Associate Professor in the School of Computer Engineering, at Nanyang Technological University, Singapore. He received B.E. from King Mongkut's Institute of Technology Ladkrabang (KMITL) in 1999. He obtained his Ph.D. in Electrical and Computer Engineering from the University of Manitoba, Canada in 2008. His research interests are in the area of radio resource management in cognitive radio networks and energy harvesting for wireless communication.

Hwee-Pink TAN (S'00–M'04–SM'14) is currently an Associate Professor of Information Systems (Practice) at the Singapore Management University (SMU). He also holds the concurrent appointment of Academic Director of the SMU-TCS iCity Lab at SMU, where he leads a team of 9 technology and social science researchers to bring together Internet of Things technologies, and social-behavioural research to enable and sustain ageing-in-place, leading, in a broader sense, to intelligent and inclusive societies, in close partnership with A*STAR, TCS, various government agencies, as well as Voluntary Welfare Organizations. Prior to joining SMU in March 2015, he spent 7 years at the Institute for Infocomm Research (I2R), A*STAR, where he was a Senior Scientist and concurrently the SERC Programme Manager for the A*STAR Sense and Sense-abilities Program. In this programme, he led a team of 30 full-time research scientists and engineers to design, pilot and evaluate architectures to support large scale and heterogeneous sensor systems to enable Smart City applications. In recognition of his contributions, he was awarded the I2R Role Model Award in 2012 and 2013, and the A*STAR Most Inspiring Mentor award, TALENT and Borderless Award in 2014.

He graduated from the Technion, Israel Institute of Technology, Israel in August 2004 with a Ph.D. In December 2004, he was awarded the A*STAR International Postdoctoral Fellowship. From December 2004 to June 2006, he was a post-doctoral research at EURANDOM, Eindhoven University of Technology, The Netherlands. He was a research fellow with The Telecommunications Research Centre (CTVR), Trinity College Dublin, Ireland between July 2006 and March 2008. His research has focused on the design, modeling and performance evaluation of underwater acoustic sensor networks, wireless sensor networks powered by ambient energy harvesting as well as large scale and heterogeneous sensor networks. He is a Senior Member of the IEEE, has published more than 100 papers, has served on executive roles for various conferences on wireless sensor networks, and is an Area Editor of the Elsevier Journal of Computer Networks.