Robust IoT Time Series Classification with Data Compression and Deep Learning

Joseph Azar[a,*], Abdallah Makhoul[a], Raphaël Couturier[a], Jacques Demerjian[b]

[a]*FEMTO-ST Institute, UMR 6174 CNRS, Univ. Bourgogne Franche-Comté, France*
[b]*LaRRIS, Faculty of Sciences, Lebanese University, Fanar, Lebanon*

Abstract

Internet of Things (IoT) and wearable systems are very resource limited in terms of power, memory, bandwidth and processor performance. Sensor time series compression can be regarded as a direct way to use memory and bandwidth resources efficiently. On the other hand, the time series classification has recently attracted great attention and has found numerous potential uses in areas such as finance, industry and healthcare. This paper investigates the effect of lossy compression techniques on the time series classification task using deep neural networks. Furthermore, this paper proposes an efficient compression approach for univariate and multivariate time series that combines the lifting implementation of the discrete wavelet transform with an error-bound compressor, namely *Squeeze* (SZ), to attain an optimal trade-off between data compression and data quality.

Keywords: IoT applications, Energy efficiency, Lossy compression, Data reduction, Time series classification, Deep neural networks, Discrete wavelet transform, lifting scheme

*Corresponding author

Email addresses: `joseph.azar@univ-fcomte.fr` (Joseph Azar),
`abdallah.makhoul@univ-fcomte.fr` (Abdallah Makhoul),
`raphael.couturier@univ-fcomte.fr` (Raphaël Couturier), `jacques.demerjian@ul.edu.lb`
(Jacques Demerjian)

1. Introduction

The world is becoming increasingly connected and intelligent through the Internet of Things (IoT), which can be defined as a wireless network of smart sensing devices connected to the Internet, ready to collect and transmit data supported by embedded devices or sensor networks. IoT devices are generally composed of five main components: 1) sensors, 2) processor, 3) memory, 4) communication module, and 5) battery. Many of these devices are connected by a gateway functionality that is capable of communicating with the sensors and providing storage and processing capabilities. This gateway could be located in the cloud or at the edge.

Time series data collected from IoT devices such as sensors, machines, and wearables, are becoming the most widespread. These data are generated with high velocity from different real-world applications such as healthcare, manufacturing, agriculture, and urban infrastructure [1] [2], and transmitted to cloud or edge processing.

One of the main issues with IoT applications is the huge volume of collected data. The on-board signal processing and compression algorithms could be used to avoid this problem and therefore avoid the transmission and storage of large amounts of data. Generally, radio communication dominates energy consumption. Therefore, data compression can significantly decrease communication energy costs by reducing the number of bits to be transmitted and thus increase the lifetime of the device [3]. Consequently, this directly affects the energy consumption and battery life of the devices in addition to the storage requirements of the transmitted data, enabling much less data to be managed.

Data compression algorithms can be divided into two types: lossless and lossy compression. Lossless compression does not involve data loss, while lossy compression leads to data loss. However, the maximum level of compression that can be achieved without loss is limited. This limited compression ratio is unfortunately a big drawback in the context of IoT data and resource constrained devices covered by this paper. On the other hand, lossy techniques cannot re-

construct the original data from the compressed data. This does not mean, however, that reconstructed data are of low quality. A good lossy compressor maintains the critical information while dropping, for example, the relatively useless data such as noise.

While on-board lossy compression techniques can increase the lifetime of IoT devices and enormously reduce the amount of data sent to the gateway, the impact of compression ratios on data analysis can not be neglected. The time series collected from IoT applications are transferred to the edge or cloud for processing. Time Series Classification (TSC) is a major and challenging problem for mining received data. TSC research have found a large number of potential applications in areas such as finance, health care, and industry. Deep Neural Networks (DNNs) are among the promising techniques used recently for TSC [4]. The main advantage of end-to-end DNN models over state-of-the-art approaches is their lack of heavy preprocessing on raw data.

In an IoT Big Data architecture, time series data pass through different phases. An example of IoT Big Data architecture can be composed of four layers as shown in Figure 1 [5]. In the first layer, time series data are generated from different IoT sources and then collected and compressed prior to transmission. The second layer provides communication between IoT devices and gateways using different communication technologies. The third layer is the main layer of the entire analytical system that processes the data using tools such as Apache Storm and Spark. In this layer, the classification of time series data takes place. The fourth layer is responsible for the interpretation and use of the generated results. In real world applications, time series data are highly compressed in layer 1 before their processing in layer 3.

Most of the proposed compression techniques for IoT in the state-of-the-art focus on compression ratio, distortion level, and energy consumption in the first layer and ignore classifier performance in the third layer. This paper focuses on time series compression and classification tasks, and studies the impact of the compression task on classification task in IoT applications. The stated problem in this work can be formulated as follow: *How does the use of time series lossy*
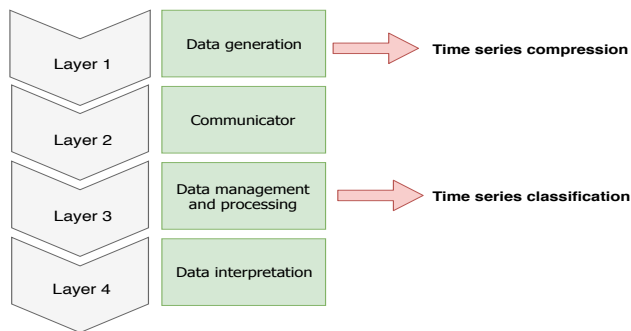
Figure 1: Example of IoT Big Data architecture

compressors on IoT nodes affect the classification performance of deep learning models?

Given the severe resource constraints of IoT nodes, the following metrics must be considered by a compression technique targeting IoT application characteristics: (1) Transmission saved energy must be higher than the energy consumed for processing. (2) Limited on-chip memories are used by embedded processors on IoT nodes. (3) The compression algorithm can be adapted to any application, sensor or activity. (4) Suitable for near real-time applications. (5) Easy to deploy and adapt to various nodes. (6) High compression ratio capability while maintaining data quality. (7) Ability to handle multi-sensor readings on one device.

One of the main contributions in this paper is the comparison of three well-used lossy compression techniques in the literature, namely compressed sensing, discrete wavelet transform, and error-bounded compression in terms of compression ratio and classification performance impact.

On the basis of this comparison, this paper proposes a new approach that combines advantageous properties of the discrete wavelet transform with the *Squeeze* (SZ) compressor, taking into account all the above-mentioned metrics and requirements.

The remainder of this paper is as follows. Related work in time series compression and classification are reported in section 2. Deep learning algorithms

used for time series classification are summarized in section 3. The compression techniques used in this paper are presented in section 4. Section 5 details the proposed compression scheme. Experimental results are detailed in section 6 and discussed in section 7, and the conclusion is presented in section 8.

2. Related work

Different data reduction techniques are proposed in the literature to resolve energy constraints [3][6]. In a recent work, Blalock *et al.* [7] presented an efficient high-ratio lossless compression technique for multivariate integer time series, namely Sprintz, with a significantly lower memory and latency than state of the art methods. Compared to different compressors, like SIMD-BP128 [8], FastP-FOR [8], Zstandard (Zstd) [9] and Zlib [10], the proposed algorithm achieved strong compression ratio, speed and memory requirements. The authors in [11] proposed a dynamic lossy compression approach to extract valuable information from IoT data with constant adaptation and information loss. The proposed technique is based on the lossy Chebyshev compression and yields a compression ratio of up to 3:1. Various compression algorithms have been proposed that take advantage of the temporal correlation in the data. A new principle for delta compression was developed by the authors in [12], which allows the compression of tempo-related data and results in a higher compression ratio and a lower memory capacity than standard delta coding. Dictionary-based compressors have been proposed to benefit from repetitive patterns in the data. In [13], the authors proposed a lossless data compression technique called Differential-Lempel-Ziv-Welch (D-LZW), which consists in finding the difference between two successive data samples before placing the data in the LZW algorithm. The authors compared their approach to bzip2 and gzip, showing better results in compressing electrocardiogram (ECG) data.

With regard to the problem stated in this work, the influence of data reduction on classification performance was investigated in previous studies. In [14], the authors examined the effect of compressed sensing and reconstruction algo-

rithms like Basis Pursuit and Orthogonal Matching Pursuit on ECG arrhythmia detection using Support Vector Machine (SVM) classifier. The results show that the classification performance remains stable until a compression ratio of 7:1, after which Basis Pursuit outperforms Orthogonal Matching Pursuit on ECG arrhythmia detection. In a similar work [15], the authors proposed the compressed sensing of time series for the recognition of human activities. The results show that the use of compressed sensing on the original time series increased the accuracy of classification using Hidden Markov Model from 77% to 95%. The explanation for this increase in classification accuracy is that the compressed sensing has some kind of advantageous dimensionality reduction, which facilitates pattern recognition. In web page classification, the authors in [16] used the idea of data reduction to make both summarized and pure data available as inputs for the classifier. Their study showed that human or machine summaries help improve the performance of the web page classification. In [17], the authors used a lossy error-bounded compressor to reduce the amount of data transmitted to the edge node. Their results showed that the compression ratio did not affect the classification accuracy of medical data, thus increasing the lifetime of IoT devices and maintaining data quality.

In the following sections, the influence of three compression techniques, namely Compressed Sensing (CS), Discrete Wavelet Transform (DWT), and Squeeze (SZ) proposed in [18], on time series classification is studied. Moreover, the proposed data compression approach is presented.

3. TSC using deep learning

Because time series data have to be properly classified, researchers have proposed numerous approaches based on deep learning models to solve this task. An empirical study of the latest DNN architectures for TSC was conducted in [19], which examines the current state of the art of deep learning algorithms for TSC. Although many kinds of DNNs exist, two principal DNN architectures used for TSC have been used in this paper: Convolutional Neural Network

(CNN), and Recurrent Neural Network (RNN).

3.1. Convolutional Neural Network

CNNs [20] are prevalent to image processing activities, reducing the number of parameters that need to be learned when the number of neuron connections in the hidden layer is limited to just certain input neurons (i.e. the local area in the image input). In many practical applications, CNNs have achieved positive results, particularly in the field of computer vision. In case of TSC problems, it is possible for CNNs to learn or automatically extract features from raw input data. A sequence of observations can be processed as a one-dimensional image, that can be read and distilled in the most salient elements by a CNN model.

This paper considers two convolutional network architectures: Fully Convolutional Networks (FCN), and Residual Network (ResNet). Both of these architectures have been tested in [21] and attained premium performance to the state-of-the-art TSC approaches.
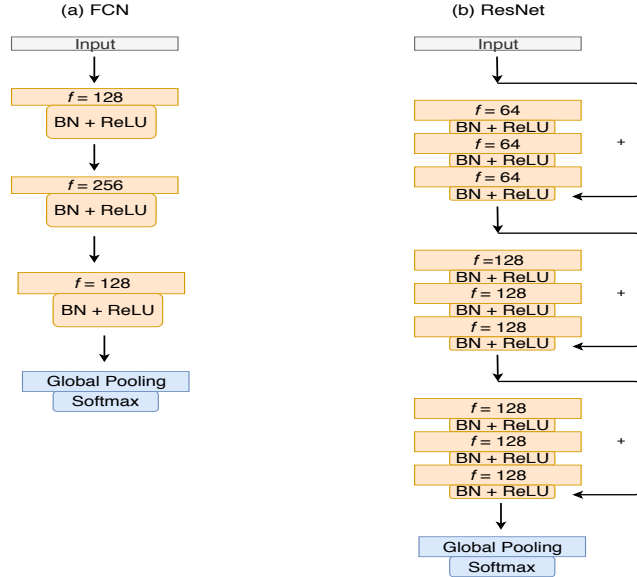

Figure 2: The network structure of FCN and ResNet tested in [21]

For semantic segmentation on images, FCN [24] has demonstrated impressive
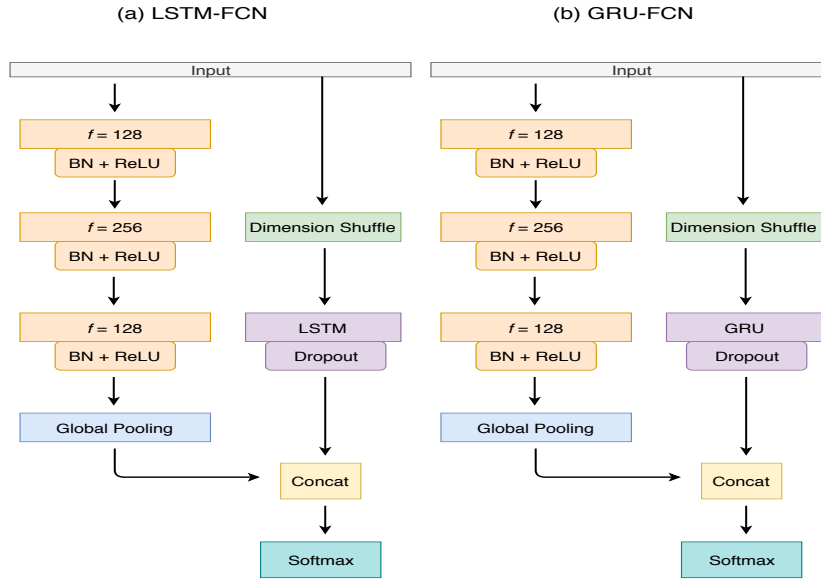
7

(a) LSTM-FCN   (b) GRU-FCN

Figure 3: The network structure of LSTM-FCN and GRU-FCN proposed in [22] and [23] respectively

quality and efficiency. The FCN used for TSC is constructed by stacking three blocks, each of which consists of a convolutional layer with f filters, followed by a batch normalization layer and a ReLU activation layer as shown in Figure 2-a. The features are fed into the global average pooling layer after the first three convolutional blocks, largely reducing the number of weights and the final result comes from the softmax layer.

ResNet is a deep CNN that uses shortcut connections between successive convolutional layers for training. The architecture used for TSC consists of 9 convolutional layers, followed by a general average pooling layer as shown in Figure 2-b.

3.2. Recurrent Neural Network

RNN is a network that has loops. The Long Short-Term Memory, or LSTM, is a type of RNN. In the context of deep learning, LSTM has recently been widely used as it is free from the problem of vanishing gradients and offers excellent

performances and results. LSTM based networks are suitable for classifying time sequences and replacing many conventional approaches to deep learning.

LSTM-FCN architecture has been proposed in [22]. The authors improved the previously discussed FCN's performance by transferring the time-series into a shuffle layer and subsequently passing it into a LSTM block as shown in Figure 3-a. The output of the LSTM block is concatenated with the output of the global pooling layer and the final results are obtained from the softmax layer.

Similarly, the authors in [23] proposed the GRU-FCN architecture, shown in Figure 3-b. The proposed architecture has replaced the LSTM with a Gated Recurrent Unit (GRU), which produces higher classification accuracy.

4. IoT Time Series Compression

In order to study the impact of data compression on the performance of DNNs for univariate TSC, three compression techniques were considered in this paper: Compressed Sensing (CS), Discrete Wavelet Transform (DWT), and an error-bounded lossy compressor, namely SZ. This section describes these compression techniques.

4.1. Compressed sensing

Compressed sensing is a digital signal processing technique capable of effectively acquiring and rebuilding a signal from a smaller number of measurements. CS can capture and display sparse signals at a rate that is considerably lower than the one normally used in the Shannon sampling theorem.

The basic measurement model for compressed sensing is defined as in Equation 1:

$$[Y]_{M,1} = [\phi]_{M,N}[X]_{N,1}, \tag{1}$$

where X is the original sparse signal of length N, Y is the compressed signal of length $M(M << N)$, and $\phi \in \mathbb{R}^{M \times N}$ is the sensing matrix that enables the reconstruction of the signal.

9

Most signals generally have a low sparsity in the time-domain representation. The representations of these signals are therefore sparse in the frequency domain or under a certain basis. In that case, X can be represented using ψ as a sparse vector as shown in Equation 2:

$$[X]_{N,1} = [\psi]_{N,N}[z]_{N,1},\tag{2}$$

where z is the frequency/spectral domain representation of the original signal X, and ψ is the matrix that converts the signal to the temporal-domain from the spectral-domain. The reconstruction of the signal allows z to be calculated from Y by combining Equation 1 and Equation 2 as follows:

$$[Y]_{M,1} = [\phi]_{M,N}[\psi]_{N,N}[z]_{N,1}.\tag{3}$$

Note that in this paper, ψ is calculated by applying the inverse Discrete Cosine Transform upon the columns of the identity matrix. Additionally, the l_1-norm minimization [25], also known as Basis Pursuit, has been employed to recover the original signal X.

This paper assumes that the data collected are time series of floating point numbers, so the compressed vector Y contains floating point data. In that case, to encode the resulting Y vector, an additional lossless entropy coder is required. This paper utilizes the lossless FPZIP compressor proposed in [26] for the encoding step.

4.2. Discrete Wavelet Transform

DWT has many advantageous properties that are valuable for time series data mining [27]. It transforms and analyzes a time series at multiple resolutions using the so-called wavelets.

The DWT allows a time-frequency representation of a signal. It splits the signal into low-frequency components (approximations) and high-frequency components (details) through the use of filters. The motivation for exploiting the DWT lies in the transformation of redundant samples in the temporal domain to decorrelated coefficients in the time-frequency domain, which allows the compaction

of the original samples and their representation with fewer coefficients. Hence, this process facilitates the study of certain features of the original dataset. The application cases of the DWT for time series analysis are excessively various, for example, compression and noise filtering [28][29].

In this paper, the DWT is used for compression by maintaining only a signal's approximation data. These approximations are floating point values that need to be encoded, similar to the compressed sensing method, the FPZIP was considered.

4.3. Error-bounded lossy compression

The third compression technique used in this article is a fast error-bounded lossy compressor namely SZ, proposed in [18] for High Performance Computing (HPC) applications. This compression scheme is intended to handle the huge quantities of data generated during the implementation of HPC applications.

The SZ technique is based on a prediction method, which corresponds to the curve fitting designed specifically to comply with error limitations specified by the user. The SZ compression requires three main steps: linearization of the input array, adaptive curve fitting and compression of unpredictable data. Three prediction models are used in the curve fitting step, based on the neighbouring data values: constant, linear and quadratic. These three models differ by the number of data points needed to fit the original value. The accepted model is that which provides the closest approximation. When no model complies with the predetermined error bound, the data point is denoted as unpredictable and then encoded by binary representation analysis. The Huffman tree is also used to encode the data fitted in the curve fitting step, which are converted into integer quantization factors.

5. Proposed compression approach

In this section, a compression scheme that enables a high compression ratio, denoises input data, and controls compression/decompression error is presented.

It begins by presenting the lifting scheme implementation of the DWT and its capacity to compress and denoise the input signal, and then details the proposed compression scheme.

5.1. Faster implementation of the DWT

The conventional method of wavelet transform needs complicated mathematical calculations, making it unfit for resource-constraining devices. Sweldens had proposed the lifting scheme in [30], a more efficient algorithm for calculating wavelet transforms and building bi-orthogonal wavelets.

The lifting scheme forward transform consists of three operations: split, predict, and update. The split operation divides the signal into even and odd samples. This step is also known as the lazy wavelet transform. In view of the locally correlated structure of the signal, even and odd samples are strongly correlated. The predict step predicts odd samples from even samples. Finally, The update step guarantees that the smoothed approximation signal has the same average as the initial signal. For a detailed explanation of the lifting procedure, refer to [29].

There are many benefits of applying the lifting scheme to IoT devices: (1) It enables a faster implementation of the wavelet transform and is very appealing for real time applications with low energy. (2) It is memory efficient and enables a fully in-place calculation. (3) Very easy to understand, implement, deploy and adapt to various nodes.

5.2. The lifting scheme for noise filtering

The data collected from real world applications are usually noisy. Data reduction is the primary reason to implement the DWT. Equally important, eliminating the large variations of data mixed in the collected signal before transmission can be a crucial step in real-time applications as this step is no longer required at the edge of the network. Thus, taking less time to process and classify the transmitted data at the edge/sink.
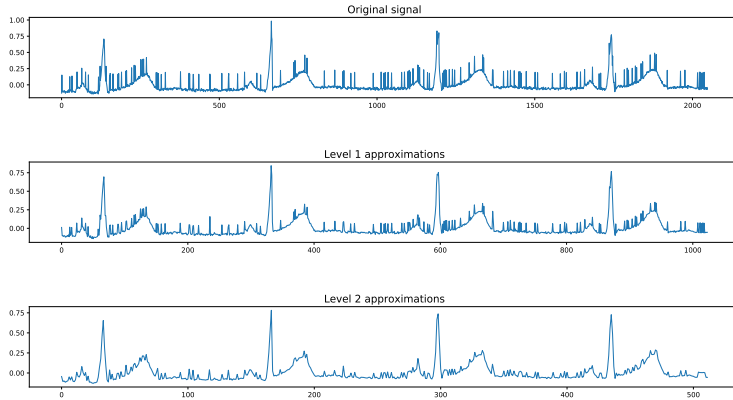
Figure 4: Example of two level decomposition of a noisy ECG signal

Figure 4 shows an example of two level decomposition of an ECG signal. In the first decomposition, the initial signal is divided into approximations and details. The approximations maintain the information of the initial signal, and the details contain the fast varying changes that may be regarded as noise. The second decomposition reapplies the lifting forward transform on the approximations resulting in level two approximations with fewer data points and less noise.

This paper proposes to apply the classification methods on the approximations of the original data. This is driven by the reality that approximations are smaller in size than the original raw data, leading in quicker implementation of the algorithms and reducing data access time. In addition, the DWT's noise filtering capacity isolates important features from the original signal. The meaningful data is thus fed to the deep learning model.

5.3. Proposed compression scheme

The data collected from sensors in IoT applications are generally multivariate time series. An M-dimensional multivariate time series, $X = [X_1, X_2, ..., X_M]$ comprises of M different univariate time series. In our compression scheme, the
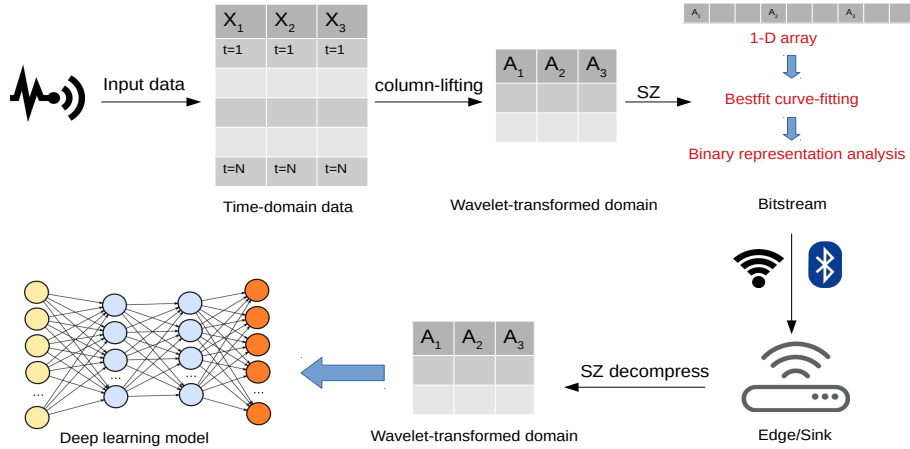
13

Figure 5: Proposed compression scheme for IoT time series classification

first step is to transform each univariate time series into approximation and detail sets using the lifting scheme. This step is referred to as column-lifting. Then the details are removed, maintaining only an M-dimensional set of approximations $A = [A_1, A_2, ..., A_M]$. Note that the lifting scheme was implemented based on the Haar wavelet due to its simplicity. Multiple levels of transformation can be done by reapplying the wavelet function on the approximation coefficients. The number of levels can be defined by the user depending on the application and the collected data.

Next, the SZ algorithm processes the approximations array. The motive behind the combination of SZ and the lifting scheme is that SZ is a prediction method, and how smooth or jagged the time series affects its output. The benefit of the lifting scheme transformation is that it results in a smoother version of the initial time series that allows the SZ's bestfit curve-fitting to better predict the data in the user-required error. When the data reaches the edge/sink, the time series reconstruction method consists of using the SZ algorithm to decompress the compressed approximation coefficients. Then the wavelet-transform data is introduced into a deep learning model without the need to apply the inverse wavelet transform and recover the original data. This process is illus-

trated in Figure 5 and Algorithm 1. The input multidimensional array is firstly processed by the forward lifting transform. Based on the provided number of decomposition levels L, the three operations (split, predict, and update) are repeated L times. The results of this operation are one matrix of approximation coefficients A and a set of L matrices of detail coefficients D $\{d_1, ..., d_L\}$. The proposed approach consists of discarding the set D, considered as noise, and keeping the approximations only. Then the approximations are converted to 1-D array, SZ's bestfit curve-fitting is applied and finally the unpredictable data points are represented using the IEEE 754 binary representation. The output is a binary array ready to be transmitted to the sink/edge.

Algorithm 1 Proposed compression approach

Require: input (N-D array), E (error bound), L (decomposition levels),

    M (num rows), N (num columns)

Ensure: *output* (binary array)

  1: output $\leftarrow$ [ ]

  2: *//forward lifting transform*

  3: **for** 1 to L **do**

  4:     split(input)

  5:     predict(input)

  6:     update(input)

  7: **end for**

  8: *//get the approximations only*

  9: A $\leftarrow$ approx(input)

10: *// SZ compression*

11: linearization(A)

12: Bestfit Curve-Fitting Compression(A, output)

13: Compressing Unpredictable Data(A,output)

5.4. Time complexity and compression performance

The time complexity of the DWT lifting scheme is $\Theta(N \log N)$, and the time complexity of the SZ algorithm is $\Theta(N)$. Figure 6 shows the execution time in seconds of the DWT, SZ, and DWT+SZ methods written in C on a Polar M600 watch for different numbers of floating-point data. The results show that the execution time of the SZ technique is low, and the proposed technique needs around 0.02 Seconds to process 2000 data points, 0.04 Seconds to process 8200 data points, and around 0.13 Seconds to process 33000 data points.
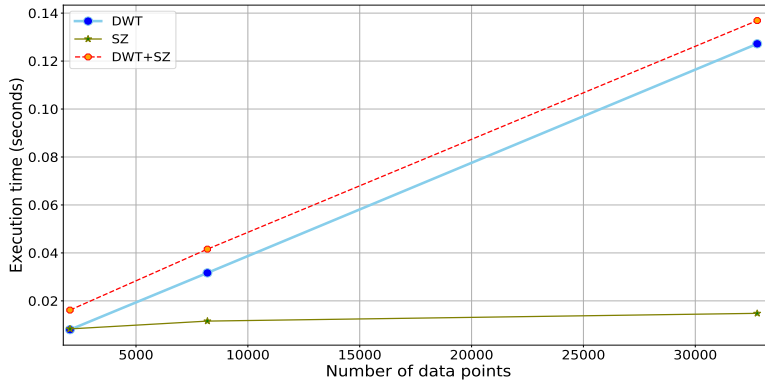

Figure 6: Execution time in seconds of the SZ, DWT, and the proposed compression scheme

In order to test how the use of the DWT can help increasing the compression ratio, 3-axis accelerometer data have been collected over 80 periods (1 period = 30 seconds). The user was at rest for the first 40 periods (sitting on a chair) and walking for the last 40 periods. Figure 7 shows the number of transmitted bytes after the compression of 36864 bytes of accelerometer data using SZ and DWT+SZ methods. The results show that the averages of the transmitted bytes per period are 742 bytes at rest and 1062 bytes at walking using SZ, and 446 bytes at rest and 718 bytes at walking using the proposed approach.
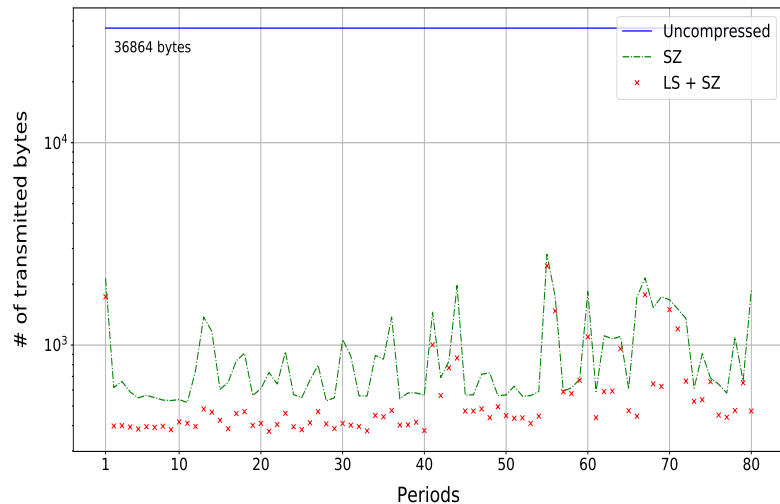
16

Figure 7: Compression of 36864 bytes of accelerometer data using SZ and DWT+SZ over 80 periods

6. Experimental Results and Analysis

The discussed compression techniques and classification models were tested on 8 UCR univariate time series datasets [31], 1 UCI multivariate time series dataset [32] and 3 UEA multivariate time series datasets [33]. Table 1 describes these time series and shows their source (ECG, Motion, Sensor, etc.), the size of the training and testing sets, the number of classification classes, the length of the time series and their type (U: univariate, M: multivariate), and the number of multivariate time series features. The results acquired on univariate and multivariate time series are presented in the followings of this section.

6.1. Selection of decomposition levels and error-bound

This section discusses the SZ error-bound value selection and the number of decomposition levels for the lifting scheme. The main goal of this paper's study is to come out with an approach that considers the trade-off between the compression ratio and the classification performance. To answer this question,

17

Dataset	Type	# Classes	Length	Train size	Test size	# Features	Type
ECG5000	ECG	5	140	500	4500	1	U
Adiac	Sinusoidal	37	176	390	391	1	U
NonInvECGTh1	ECG	42	750	1800	1965	1	U
NonInvECGTh2	ECG	42	750	1800	1965	1	U
StarLightCurves	Sensor	3	1024	1000	8236	1	U
UWaveX	Motion	8	315	896	3582	1	U
UWaveY	Motion	8	315	896	3582	1	U
UWaveZ	Motion	8	315	896	3582	1	U
HAR [32]	Motion	6	128	7352	2947	9	M
ECG [33]	ECG	2	152	100	100	2	M
UWave [33]	Motion	8	315	200	4278	3	M
Wafer [33]	Motion	2	198	298	896	6	M

Table 1: The descriptions of the utilized datasets based on [32], [33], and [31]

an empirical approach was taken, recording the model's classification accuracy trained on each dataset with different SZ error-bound and DWT decomposition level.

Figure 8 shows the outcome of the experiment on two datasets, namely Adiac and NonInvasiveFatalECG_Thorax1. For the SZ algorithm, the results show that the classification accuracy for an error bound of 10^{-3} remains very similar to that of the model trained on the original dataset, but the compression ratio is low compared to other compression settings. The highest compression ratio was achieved for an error bound of 10^{-1}, but the classification performance dropped significantly. Using a 10^{-2} error bound resulted in a good balance between classification accuracy and compression ratio. As for the number of decomposition levels, it can be seen that doing more than two wavelet decompositions greatly reduces classification accuracy while one level decomposition keeps the accuracy similar to the one of the original model and two levels decomposition helps to improve the compression ratio at the cost of slightly sacrificing the classification performance.

In this paper the proposed approach is to combine the lifting scheme with SZ. Two combinations were considered in the followings according to the above
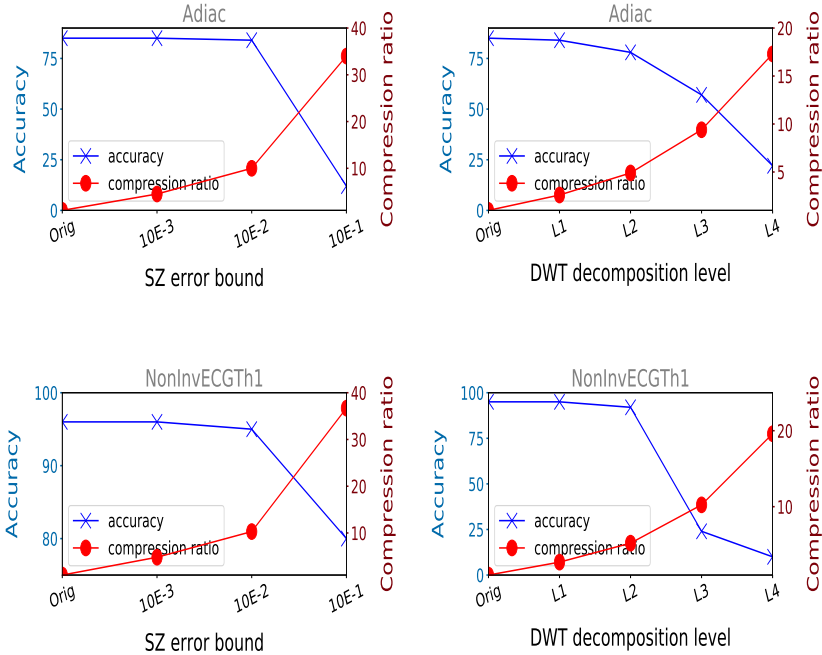
Figure 8: Accuracy vs compression ratio for different SZ error-bound values and DWT decomposition levels. The graphs correspond to Adiac and NonInvasiveFatalECG_Thorax1 datasets

experiment: DWT L1+SZ E-2 and DWT L2+SZ E-2.

6.2. Univariate time series

The first studies were conducted on univariate time series. Two metrics were considered: compression ratio and classification performance.

6.2.1. Compression ratio

The following compares the compression ratios of the techniques discussed in sections 4 and 5. Different variations of these techniques with different compression ratios were considered as follows:

- SZ E-1: SZ with an absolute error bound of 10^{-1} (the decompressed value should be within the range $[V - 10^{-1}, V + 10^{-1}]$)

- SZ E-2: SZ with an absolute error bound of 10^{-2}

- SZ E-3: SZ with an absolute error bound of 10^{-3}

- CS20: Compressed sensing with $M = \frac{N \times 20}{100}$

- CS40: Compressed sensing with $M = \frac{N \times 40}{100}$

- DWT L2: Two level DWT decomposition using Haar wavelet

- DWT L1: One level DWT decomposition using Haar wavelet

- DWT L1 + SZ E-2: One level DWT decomposition using the lifting scheme based Haar transform followed by SZ with an absolute error bound of 10^{-2}

- DWT L2 + SZ E-2: Two levels DWT decomposition using the lifting scheme based Haar transform followed by SZ with an absolute error bound of 10^{-2}

To compare the performance of the aforementioned techniques, the compression ratio (CR) is calculated as follows:

$$CR = \frac{original\_data}{compressed\_data} \tag{4}$$

	ECG5000	Adiac	NonInv-ECGTh1	NonInv-ECGTh2	StarLight-Curves	UWaveX	UWaveY	UWaveZ
SZ E-1	14.73	34.37	36.73	37.91	69.82	32.97	35.07	33.97
SZ E-2	5.83	10.18	10.33	10.77	29.34	10.48	11.40	10.91
SZ E-3	3.07	4.58	4.81	4.90	10.85	5.61	6.12	5.63
CS40	3.12	3.2	3.43	3.46	3.73	4.11	4.34	4
CS20	6	6.12	6.62	6.68	7.14	7.57	7.97	7.54
DWT L2	4.8	4.9	5.28	5.33	5.71	5.31	5.39	5.31
DWT L1	2.52	2.58	2.74	2.77	2.99	2.92	2.98	2.91
DWT L1 + SZ E-2	9.06	13.09	13.95	14.48	29.89	13.23	13.93	13.40
DWT L2 + SZ E-2	12.3	18.8	21.2	21.7	32.7	19	19.7	19.1

Table 2: Compression ratios achieved by the different data compression techniques on the UCR univariate time series [31]

The obtained results are shown in Table 2. Table 2 shows the compression ratio of the different compression techniques. It can be seen that the best results are achieved by the SZ method. The closer the absolute error bound value is to zero, the higher the compression ratio is. As for the compressed sensing, the compression ratio depends on M, and on the number of decomposition levels in the case of DWT. Note that CS and DWT outputs are encoded using FPZIP. As for the proposed approach, it can be seen that combining the SZ method with an appropriate error-bound and the DWT helps to achieve a higher compression ratio. For instance, SZ with an absolute error of 10^{-2} achieves a compression ratio of 5:1 on the ECG5000 dataset, while the compression ratio has been increased to 9:1 and 12:1 when combining SZ with DWT with one level decomposition and two levels decomposition respectively. The reason behind the improvement in the performance resides in the fact that each decomposition level of the DWT results in a more smoothed and correlated approximation coefficients. As a result, the SZ computes the approximation coefficients instead of all the data points in a time series. Furthermore, the DWT removes the noises from a time series, allowing the SZ method to perform better and to produce better results.

6.2.2. Classification performance

This section compares the four DNN models' classification performances on the compressed univariate time series. The considered experimental adjustments for these models are shown in Table 3.

In this paper, the datasets used contain periodic and non-periodic time series. Periodic time series are like ECG and sinusoidal data, and non-periodic time series are like motion data. Each of the four DNN models was applied 5 times on the different compressed versions of the time series and the average accuracy was recorded. Figures 9 and 10 represent each time series in a radar chart showing the accuracy metric obtained by each of the four NN models implemented with Keras on the compressed variants of the time series. Note that Figure 9 shows the periodic univariate time series and Figure 10 shows the

	Resnet		LSTM FCN		GRU FCN		FCN	
	Epochs	Batch	Epochs	Batch	Epochs	Batch	Epochs	Batch
ECG5000	200	16	400	128	400	128	200	128
Adiac	1500	16	4500	128	4500	128	1500	128
NonInvECGTh1	1500	16	4500	128	4500	128	1500	128
NonInvECGTh2	1500	16	4500	128	4500	128	1500	128
StarLightCurves	500	16	1000	128	1000	128	500	128
UWaveX	750	16	2000	128	2000	128	750	16
UWaveY	750	16	2000	128	2000	128	750	16
UWaveZ	750	16	2000	128	2000	128	750	16

Table 3: Experimental adjustments used in the implementation of the four DNN models

non-periodic univariate time series.

It can be seen from these figures that the SZ method with an absolute error of 10^{-1} does not maintain the quality of the time series. Although it has the highest compression ratio, the classification accuracy of most datasets has been greatly affected. For example, the maximum classification accuracy values obtained on the uncompressed Adiac dataset and the compressed Adiac dataset with SZ E-1 (Figure 9) are 0.83 and 0.64 respectively. This can show the effect of data compression on the classification task. The SZ with absolute errors of 10^{-2} and 10^{-3} slightly affected the performance of the DNN models.

As for the CS method, the reconstruction quality of the compressed data was not satisfactory. In spite of the fact that the CS method is known to reconstruct a signal perfectly, the signal must be S-sparse, meaning noise-free, in order to be perfectly reconstructed, which is not the case for the time series used and the most real life data collected from IoT devices. This explains the performance decrease of the classifiers when working on reconstructed time series using CS. It can be noticed that the classification performance of the DNN models is highly affected by parameter M. For example, on the uncompressed Adiac dataset, the maximum accuracy achieved is 0.84 versus 0.66 on the reconstructed dataset from 40% of the original data points (CS 40) and 0.4 on the reconstructed dataset from 20% of the original data points (CS 20). On the other hand, the classification performance of the models was slightly affected
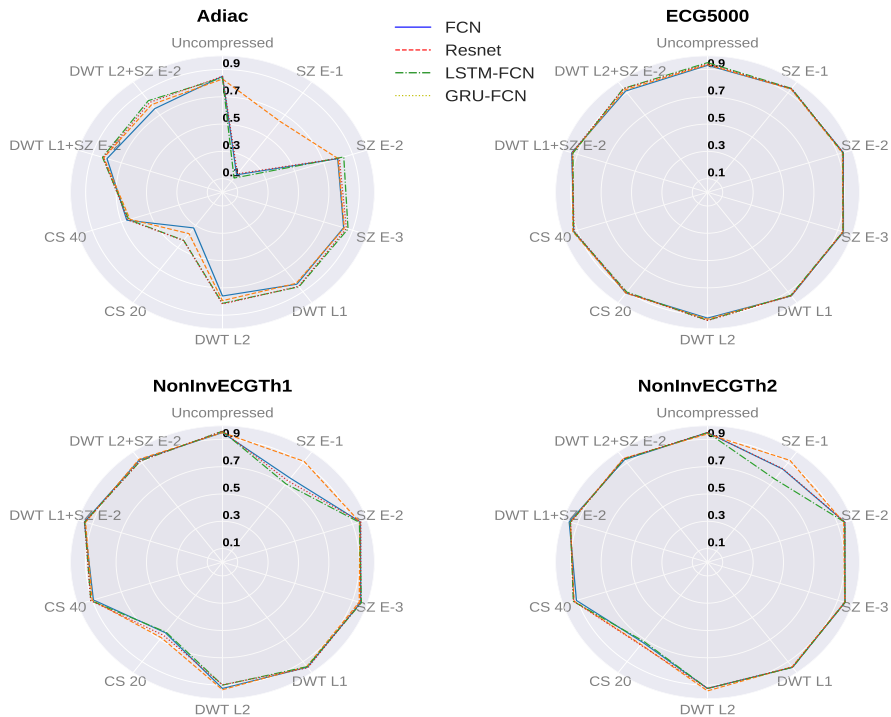
Figure 9: Accuracy metrics achieved by the four DNN models on the compressed periodic univariate time series

by the DWT method, which was able to maintain the quality of the data. This can be explained by the denoising ability of the DWT method. The results also show that the classification accuracy values of the time series compressed with the proposed approach (DWT L1 + SZ E-2 and DWT L2 + SZ E-2) remained very close to those of the uncompressed time series. Taking into account the trade-off between compression ratio and classification performance, the proposed approach has yielded the best results on the univariate time series. It is important to note that the performance of the Resnet model was the most robust against compression in the experiments conducted. This can be seen in Figures 9 and 10, where data is compressed with SZ E-1 and CS 20, Resnet has produced the best accuracy compared to other models.
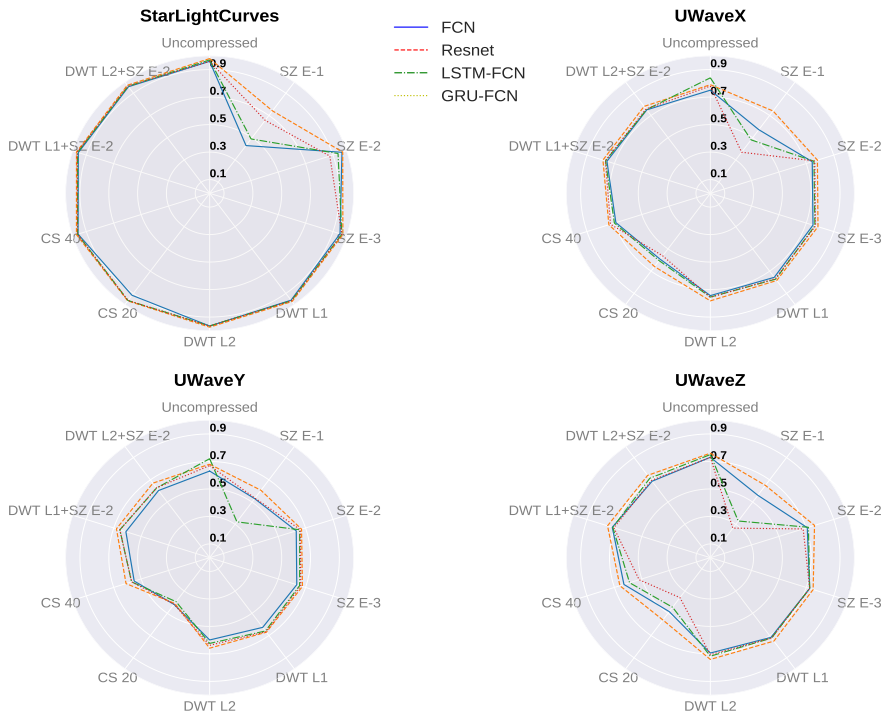
23

Figure 10: Accuracy metrics achieved by the four DNN models on the compressed non-periodic univariate time series

6.3. Multivariate time series

The compression approach presented can handle both univariate time series and multivariate time series. The followings address the effectiveness of the proposed approach when dealing with multivariate time series in terms of compression ratio, classification performance, and energy efficiency.

6.3.1. Compression ratio

The compression ratio of the different compression techniques achieved on the 3 UEA multivariate time series datasets in addition to the reconstruction time in seconds are shown in Table 4. Compared to the results obtained on the univariate time series, SZ with 10^{-1} error-bound and the two combinations of DWT and SZ achieved the highest compression ratio. However, in the case of multivariate time series, DWT L2+SZ E-2 outperformed SZ E-1 for UWave

and Wafer datasets, and the compression performance of DWT L1+SZ E-2 became closer to that of SZ E-1. As for the reconstruction time, for the compressed sensing technique, the time needed to reconstruct the compressed data increases with the number of features in a dataset. This can be explained by the fact that CS uses a method of optimization to recover the data, and the more features a dataset has, the greater the process of minimizing the $l1$-norm. The SZ method's reconstruction time ranged from 0.02 to 0.05 seconds, making it very suitable for applications requiring quick decision-making. As for the DWT, the reconstruction time was not calculated since this paper's proposed solution is to use the approximation coefficients directly for classification without reconstruction of the original data.

	ECG		UWave		Wafer	
	CR	rec time (sec)	CR	rec time (sec)	CR	rec time (sec)
SZ E-1	27	0.02	6.46	0.04	12.9	0.05
SZ E-2	5.4	0.02	3.36	0.03	4.4	0.05
SZ E-3	2.5	0.02	2	0.03	2.66	0.05
CS 40	2.8	15.4	2.7	255	2.64	280.2
CS 20	5.35	27.4	5.42	300.6	5.29	308.7
DWT L1	4.6	-	5.37	-	5.4	-
DWT L2	2.4	-	3	-	2.9	-
DWT L1+SZ E-2	12.6	0.02	6	0.03	8.37	0.05
DWT L2+SZ E-2	19.6	0.02	11.2	0.03	15.2	0.05

Table 4: Compression ratios and reconstruction time (seconds) achieved by the different data compression techniques on the UEA multivariate time series [33]

Another experiment was carried out in which 300 batches of 1000 feature vectors were taken from the HAR [32] dataset, each containing 9 features, and deployed on a Polar M600 wearable. We considered that each batch corresponds to a period of time. The techniques described in the previous section compress the corresponding batch for each period and the resulting number of bytes is recorded. After 300 periods, the average number of output bytes for each com-

pression method is calculated with the standard deviation. The obtained results are shown in Figure 11
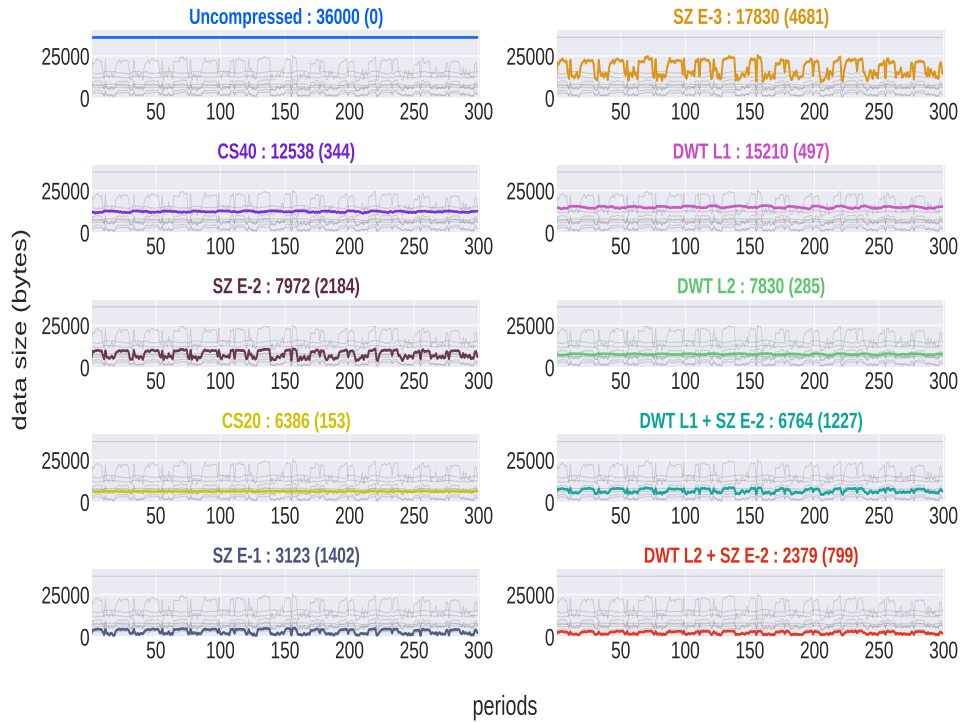

Figure 11: Compression of $[1000 \times 9]$ batches of the HAR [32] dataset over 300 periods. Next to each compression technique is shown the average number of bytes per period alongside the standard deviation (shown in parenthesis).

Figure 11 indicates two main things: (1) the combination of DWT and SZ improved the compression performance, (2) the standard deviation of the number of bytes after compression at each period is high for SZ relative to CS and DWT. The latter can be explained by the fact that the compressed data are motion data (accelerometer and gyroscope), and these time series are generally jagged and vary rapidly between different activities (sitting, walking, running, etc). Since SZ is based on a prediction mechanism, its performance depends heavily on the characteristics and variability of the data. For the same reason,

using DWT increased the performance of SZ due to each level of decomposition resulting in a smoother version of the data.

6.3.2. Classification performance

The FCN model was used to evaluate the effectiveness of our compression approach on the multivariate datasets. The data processed by the model are tensors with the dimensions (samples, seq_len, n_channels), where samples is the number of training examples, seq_len is the number of steps in the time series, and n_channels is the number of columns (features) in the data.

The FCN model was trained on three variants of the data. In the first one, the model processes the original data. In the second and third variants, the model processes the approximations resulting from one level decomposition $(seq\_len = \frac{length}{2})$ and two levels decomposition $(seq\_len = \frac{length}{4})$ using the column lifting. Note that the length of each dataset is shown in Table 1.

Dataset	Input data	Length	Accuracy
ECG	original	152	0.87
	level 1 approximations	76	0.86
	level 2 approximations	38	0.72
UWave	original	315	0.93
	level 1 approximations	157	0.93
	level 2 approximations	78	0.80
Wafer	original	198	0.98
	level 1 approximations	99	0.98
	level 2 approximations	49	0.89
HAR	original	128	0.93
	level 1 approximations	64	0.93
	level 2 approximations	32	0.92

Table 5: Classification accuracy achieved by the FCN model on the multivariate time series datasets

Table 5 shows the classification accuracy achieved by the FCN on the four datasets with different input data. The results show that the model was able

27

to learn from compressed data. The results also show that training on level-2 approximations decreased classification performance compared to training on level-1 approximations for ECG, UWave, and Wafer datasets. The model, on the other hand, learned well from the HAR dataset's level-2 approximations. The amount of training data will explain this discrepancy. The training size of the ECG, UWave, and Wafer datasets ranges from 100 to 300, while the training size of the HAR dataset is 7352, as shown in Table 1. The small amount of training data did not help the model to learn from level-2 approximations the necessary structure.

6.3.3. Energy efficiency

The proposed compression scheme was implemented on a Polar M600 wearable using Android NDK toolset in order to test the impact of data compression on energy conservation. The collected data are multivariate time series consisting of 3-axis accelerometer. These time series are transmitted to a local PC using Android (Bluetooth Low Energy) BLE after each period. Figure 12 shows the wearable battery level change over 120 periods (1 period = 1 min) for four distinct situations:

- Yellow line: the wearable device is in the idle state

- Red line: the wearable device is continuously collecting data (motion sensors are turned on)

- Green line: The wearable device is continuously collecting data and compresses and transmits data after each period

- Blue line: the wearable device is continuously collecting data, and transmits data after each period (no compression)

The results show that the battery level decreased to 76% when continuously collecting data, to 72% when performing sensing, compression, and transmission, and to 56% when performing sensing and transmission. Note that we set the sensing frequency for testing purposes to the maximum in this experiment,
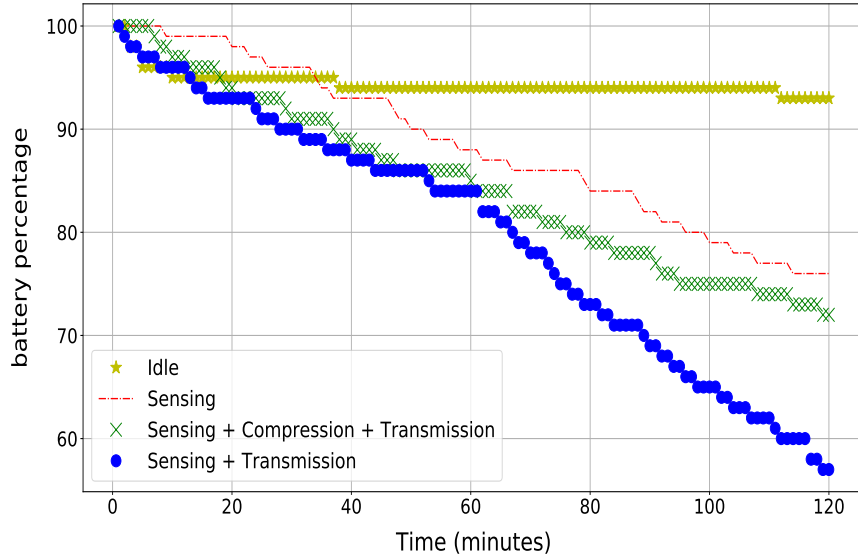
28

Figure 12: Polar M600 battery level over 120 periods for the four situations: Idle, Sensing, Sensing+compression+transmission, and sensing+transmission

generally gathering data with the default sensing frequency does not consume so much energy. From these outcomes, it can be seen that by implementing the proposed compression scheme, the device's lifetime can be improved by 16% after 2 hours.

7. Discussion

Various insights can be taken on the basis of the experiments conducted in this study. Recall that the main goal is to propose a compression approach that will help maximize an IoT device's lifetime. It operates on multivariate time series and floating point data efficiently. It can be tailored to any application, sensor or activity and suitable for near real-time applications.

The compressed sensing method with a low value of M showed a good compression ratio but after reconstruction did not maintain the required data features, thus decreasing the quality of the classification. Another weak point for

compressed sensing is the reconstruction time, which is highly dependent on the data size and the number of features. Note that compressed sensing works perfectly when the sparsity of the data is high in the time/frequency domain, which is not always the case in IoT applications. On the other hand, the SZ method yielded a high compression ratio and the classification performance remained intact with the appropriate error-bound. One concern about SZ is that its performance depends on the stationarity of the time series as shown in Figure 11.

The proposed approach of using the discrete wavelet transform as a denoising and smoothing step prior to SZ can be useful for applications where the time series collected are jagged, such as recognition of human activity where high variations can be found between consecutive data points. The proposed approach helped to increase the compression ratio and maintain classification performance when using DWT with one level decomposition. Training a neural network with two levels decomposition approximations decreased the classification accuracy by up to 13% for some datasets. Two things can be recommended to solve this problem: (1) Increasing the size of the training set or testing this approach on datasets such as HAR [31] which contain a good amount of training data enabling the model to learn from the compressed time series the necessary features. (2) Test this approach with wavelets that are more complex than the Haar wavelet. The selection of the corresponding wavelet family is considered for future work.

8. Conclusion

Considering that IoT devices are generally limited in calculation, storage and energy, data compression can be seen as an effective way to increase the lifetime of these devices. An empirical study has been conducted in this paper to identify appropriate compression ratios, which maintain important information from different time series. Different variations of compression techniques were considered based on an error-bound lossy compressor, namely SZ, the Discrete

Wavelet Transform (DWT), and the Compressed Sensing (CS). Additionally, a new compression approach that combines the DWT lifting scheme and SZ has been proposed. The proposed method can manage univariate and multivariate time series effectively, attain a high compression ratio, and can be easily implemented and tailored to various applications.

In order to study the impact of time series compression on the classification task, four newly proposed deep neural networks for time series classification (TSC) were implemented. The experimental results showed that using DWT as a transformation step followed by SZ helps to denoise input data and enables SZ to compress more smooth approximation coefficients resulting in a higher compression ratio.

In addition, this paper suggested to use the frequency-domain coefficients as an entry to the classification models, showing that deep learning models can still learn features from compressed data and attain good classification accuracy.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of things: A survey on enabling technologies, protocols, and applications, IEEE Communications Surveys Tutorials 17 (4) (2015) 2347–2376. doi:10.1109/COMST.2015.2444095.

[2] D. Helbing, D. Brockmann, T. Chadefaux, K. Donnay, U. Blanke, O. Woolley-Meza, M. Moussaid, A. Johansson, J. Krause, S. Schutte, M. Perc, Saving human lives: What complexity science and information

systems can contribute, Journal of Statistical Physics 158 (3) (2015) 735–781. `doi:10.1007/s10955-014-1024-9`.
URL `https://doi.org/10.1007/s10955-014-1024-9`

[3] M. A. Razzaque, C. Bleakley, S. Dobson, Compression in wireless sensor networks: A survey and comparative evaluation, ACM Trans. Sen. Netw. 10 (1) (2013) 5:1–5:44.

[4] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P. Muller, Deep learning for time series classification: a review, CoRR abs/1809.04356. `arXiv: 1809.04356`.
URL `http://arxiv.org/abs/1809.04356`

[5] M. M. Rathore, A. Ahmad, A. Paul, S. Rho, Urban planning and building smart cities based on the internet of things using big data analytics, Comput. Netw. 101 (C) (2016) 63–80. `doi:10.1016/j.comnet.2015.12.023`.
URL `http://dx.doi.org/10.1016/j.comnet.2015.12.023`

[6] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, K. Araki, Practical data compression in wireless sensor networks: A survey, J. Netw. Comput. Appl. 35 (1) (2012) 37–59. `doi:10.1016/j.jnca.2011.03.001`.
URL `http://dx.doi.org/10.1016/j.jnca.2011.03.001`

[7] D. Blalock, S. Madden, J. Guttag, Sprintz: Time series compression for the internet of things, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 2 (3) (2018) 93:1–93:23. `doi:10.1145/3264903`.
URL `http://doi.acm.org/10.1145/3264903`

[8] D. Lemire, L. Boytsov, Decoding billions of integers per second through vectorization, Softw. Pract. Exper. 45 (1) (2015) 1–29. `doi:10.1002/spe. 2203`.
URL `http://dx.doi.org/10.1002/spe.2203`

[9] Y. Collet, Zstandard-fast real-time compression algorithm, https://facebook.github.io/zstd/ (2017).

[10] P. Deutsch, J.-L. Gailly, Zlib compressed data format specification version 3.3 (1996).

[11] A. Ukil, S. Bandyopadhyay, A. Pal, Iot data compression: Sensor-agnostic approach, in: 2015 Data Compression Conference, 2015, pp. 303–312. doi: 10.1109/DCC.2015.66.

[12] B. R. Stojkoska, Z. Nikolovski, Data compression for energy efficient iot solutions, in: 2017 25th Telecommunication Forum (TELFOR), 2017, pp. 1–4. doi:10.1109/TELFOR.2017.8249368.

[13] T. L. Le, M. Vo, Lossless data compression algorithm to save energy in wireless sensor network, in: 2018 4th International Conference on Green Technology and Sustainable Development (GTSD), 2018, pp. 597–600. doi:10.1109/GTSD.2018.8595614.

[14] S. Zareei, J. D. Deng, Impact of compression ratio and reconstruction methods on ecg classification for e-health gadgets: A preliminary study, in: T. Mitrovic, B. Xue, X. Li (Eds.), AI 2018: Advances in Artificial Intelligence, Springer International Publishing, Cham, 2018, pp. 85–97.

[15] O. P. Concha, R. Y. D. Xu, M. Piccardi, Compressive sensing of time series for human action recognition, in: 2010 International Conference on Digital Image Computing: Techniques and Applications, 2010, pp. 454–461. doi:10.1109/DICTA.2010.83.

[16] D. Shen, Z. Chen, Q. Yang, H.-J. Zeng, B. Zhang, Y. Lu, W.-Y. Ma, Web-page classification through summarization, in: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '04, ACM, New York, NY, USA, 2004, pp. 242–249. doi:10.1145/1008992.1009035.
URL http://doi.acm.org/10.1145/1008992.1009035

[17] J. Azar, A. Makhoul, M. Barhamgi, R. Couturier, An energy efficient iot data compression approach for edge machine learn-

ing, Future Generation Computer Systems 96 (2019) 168 – 175. doi:https://doi.org/10.1016/j.future.2019.02.005.
URL http://www.sciencedirect.com/science/article/pii/S0167739X18331716

[18] S. Di, F. Cappello, Fast error-bounded lossy hpc data compression with sz, in: 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Vol. 00, 2016, pp. 730–739. doi:10.1109/IPDPS.2016.11. URL doi.ieeecomputersociety.org/10.1109/IPDPS.2016.11

[19] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P. Muller, Deep learning for time series classification: a review, CoRR abs/1809.04356. arXiv:1809.04356. URL http://arxiv.org/abs/1809.04356

[20] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324. doi:10.1109/5.726791.

[21] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: A strong baseline, CoRR abs/1611.06455. arXiv:1611.06455. URL http://arxiv.org/abs/1611.06455

[22] F. Karim, S. Majumdar, H. Darabi, S. Chen, Lstm fully convolutional networks for time series classification, IEEE Access 6 (2018) 1662–1669. doi:10.1109/ACCESS.2017.2779939.

[23] N. Elsayed, A. S. Maida, M. Bayoumi, Deep gated recurrent and convolutional network hybrid model for univariate time series classification, CoRR abs/1812.07683. arXiv:1812.07683. URL http://arxiv.org/abs/1812.07683

[24] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for seman-

tic segmentation, CoRR abs/1411.4038. `arXiv:1411.4038`.
URL `http://arxiv.org/abs/1411.4038`

[25] E. C, J. Romberg, T. Tao, Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information (2004).

[26] P. Lindstrom, M. Isenburg, Fast and efficient compression of floating-point data, IEEE Transactions on Visualization and Computer Graphics 12 (5) (2006) 1245–1250. `doi:10.1109/TVCG.2006.143`.

[27] P. Chaovalit, A. Gangopadhyay, G. Karabatis, Z. Chen, Discrete wavelet transform-based time series analysis and mining, ACM Comput. Surv. 43 (2) (2011) 6:1–6:37. `doi:10.1145/1883612.1883613`.
URL `http://doi.acm.org/10.1145/1883612.1883613`

[28] J. Azar, C. Habib, R. Darazi, A. Makhoul, J. Demerjian, Using adaptive sampling and dwt lifting scheme for efficient data reduction in wireless body sensor networks, in: 2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2018, pp. 1–8. `doi:10.1109/WiMOB.2018.8589093`.

[29] J. Azar, R. Darazi, C. Habib, A. Makhoul, J. Demerjian, Using dwt lifting scheme for lossless data compression in wireless body sensor networks, in: 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC), 2018, pp. 1465–1470. `doi:10.1109/IWCMC.2018.8450459`.

[30] W. Sweldens, The lifting scheme: A construction of second generation wavelets, SIAM Journal on Mathematical Analysis 29 (1998) 511–546. `arXiv:https://doi.org/10.1137/S0036141095289051`.

[31] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, The ucr time series classification archive, `www.cs.ucr.edu/~eamonn/time_series_data/` (July 2015).

[32] D. Anguita, A. Ghio, L. Oneto, X. Parra, J. L. Reyes-Ortiz, A public domain dataset for human activity recognition using smartphones, in: ESANN, 2013.

[33] A. J. Bagnall, H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, E. J. Keogh, The UEA multivariate time series classification archive, 2018, CoRR abs/1811.00075. arXiv:1811.00075.
URL http://arxiv.org/abs/1811.00075