An energy efficient IoT data compression approach for edge machine learning

Joseph Azar, Abdallah Makhoul, Mahmoud Barhamgi, Raphael Couturier

An energy efficient IoT data compression approach for edge machine learning

Joseph Azar[a], Abdallah Makhoul[a], Mahmoud Barhamgi[b], Raphaël Couturier[a]

[a]*FEMTO-ST Institute, UMR 6174 CNRS, Univ. Bourgogne Franche-Comté, France*
[b]*LIRIS UMR 5205 CNRS, Claude Bernard Lyon 1 University, Lyon, France*

Abstract

Many IoT systems generate a huge and varied amount of data that need to be processed and responded to in a very short time. One of the major challenges is the high energy consumption due to the transmission of data to the cloud. Edge computing allows the workload to be offloaded from the cloud at a location closer to the source of data that need to be processed while saving time, improving privacy, and reducing network traffic. In this paper, we propose an energy efficient approach for IoT data collection and analysis. First of all, we apply a fast error-bounded lossy compressor on the collected data prior to transmission, that is considered to be the greatest consumer of energy in an IoT device. In a second phase, we rebuild the transmitted data on an edge node and process it using supervised deep learning techniques. To validate our approach, we consider the context of driving behavior monitoring in intelligent vehicle systems where vital signs data are collected from the driver using a Wireless Body Sensor Network (WBSN) and wearable devices and sent to an edge node for stress level detection. The experimentation results show that the amount of transmitted data has been reduced by up to 103 times without affecting the quality of medical data and driver stress level prediction accuracy.

Keywords: IoT, Edge computing, Data compression, Machine learning, Energy efficiency, Stress detection

Email addresses: `joseph.azar@univ-fcomte.fr` (Joseph Azar), `abdallah.makhoul@univ-fcomte.fr` (Abdallah Makhoul), `mahmoud.barhamgi@univ-lyon1.fr` (Mahmoud Barhamgi), `raphael.couturier@univ-fcomte.fr` (Raphaël Couturier)

1. Introduction

Cloud computing that is centrally deployed on a global scale has become an indispensable part of processing IoT data. However, cloud-assisted Internet of things (CoT) faces several difficulties such as transmission latency, bandwidth constraints, and high energy consumption. For instance, sending a single bit of data over the cellular network consumes a lot of energy which decreases the lifetime of the IoT system.

On the other hand, edge computing has emerged as a promising paradigm that pushes the cloud services to the edge of the network. It can be seen as a decentralized cloud that drives the computing power closer to the source of data and allows local decision making. Edge computing was shown to be a better solution than the cloud in numerous IoT applications [1]. For instance, applications that demand near real-time responses such as autonomous driving cars and eHealth can not work properly with the cloud due to the high latency and ineffective bandwidth caused by the large number of sensors connected to the network.

Wireless Sensor Networks (WSNs), Wireless Body Sensor Networks (WBSNs), and wearable devices make up the essential blocks of IoT architectures. Many of these smart objects, that are responsible for the collection, processing, and transmission of data, are still battery operated and resource constrained. The three major constituents of a smart object that consume energy are the microcontroller (MCU), transceiver, and sensor units. Among all tasks, it is well known that data transmission is the highest energy-consuming task in IoT nodes [2] [3]. An important step towards energy efficiency in IoT applications is the transfer of computational tasks from the cloud to the edge. In general, the radio communication task between the IoT nodes and the edge consumes less energy than transmitting the data directly to the cloud over the cellular network [1] [4]. Equally important, reducing the amount of data to be transmitted to the edge can further increase the lifetime of the IoT nodes and save storage at the edge.

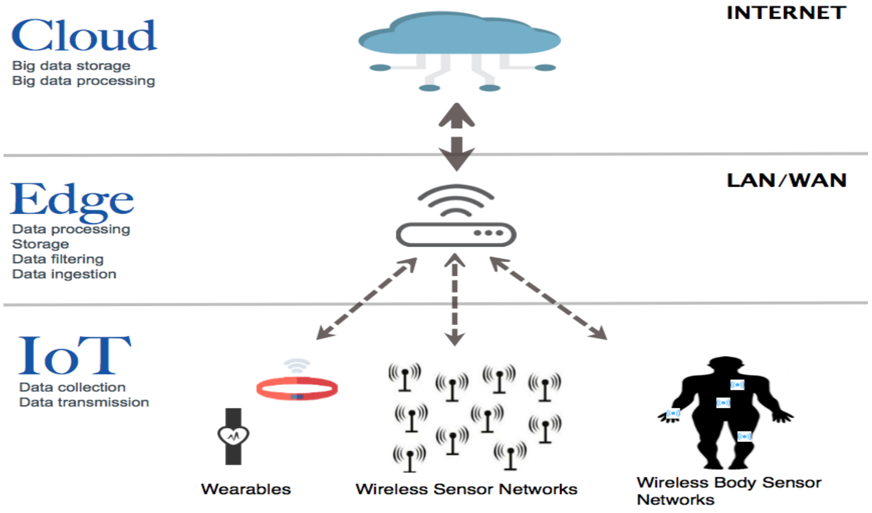

Figure 1: IoT network architecture

In this paper, we consider the IoT network architecture shown in Figure 1, where the data are collected from smart objects such as wearables and sensor devices and sent periodically to an edge node using short range communication protocols (e.g. WiFi, Bluetooth). The edge node is responsible for the processing, analyzing, filtering, storing, and sending the data to the cloud. To start, The energy conservation problem is tackled by proposing a fast error-bounded lossy data compression technique to be deployed on the IoT devices. The objective is to reduce the number of bits to be transmitted periodically to the edge node. Then, we study the effect of lossy compression on the performance of machine and deep learning models deployed at the edge, and trained on reconstructed data with degraded quality as compared to the original data. To do so, we consider the driving behavior monitoring use case where physiological signals are collected from drivers and sent to the edge node in order to detect their stress level. The stated problem in this work can be formulated as follow: *Does the loss of information due to lossy compression and energy conservation techniques deployed on the IoT nodes affect the analysis and processing of the* *data at the edge?*

The rest of the paper is organized as follows: Section 2 presents the work related to data analysis on the edge and data reduction in IoT applications. Section 3 explains the proposed compression scheme. Section 4 discusses the case studied in this paper and presents the prediction model deployed at the edge. Section 5 details the experimental results. Section 6 discusses and analyses the obtained results and section 7 concludes the paper.

2. Related Work

In this section, we first introduce the analysis of data in edge computing using machine and deep learning, and then we discuss data reduction in IoT applications.

2.1. Machine and deep learning in edge computing

The use of machine and deep learning techniques for data processing could help edge devices to be smarter, and improve privacy and bandwidth usage. In [5], the authors introduced deep learning for IoT into the edge computing environment and proposed an approach that optimizes network performance and increase user privacy. Machine and deep learning in edge computing can bring multiple improvements to the traditional approaches that rely on cloud computing by:

- Processing data using conventional machine learning techniques and transferring the results or necessary features extracted from raw sensor data.
- Deploying part of deep learning networks layers on the edge and transferring the extracted features whose size is smaller than that of the input data.
- Deploying neural networks on the edge with minimized size that maintain accuracy.

- Training the networks on the cloud and shipping the trained models to the edge.

The aforementioned approaches reduce the pressure on the network by reducing the size of the data to be transferred to the cloud. For instance, each layer in a deep learning network processes and scales down the size of the generated features from the previous layer. The more layers are deployed on the edge, the smaller is the size of the features to be transferred to the cloud and the more they are incomprehensible, hence, increasing the privacy.

Edge nodes are devices such as mobile phones, IoT gateways, and local PCs that have a limited processing capability as compared to cloud servers. The size of neural networks deployed on these devices should be reasonable. In [6], the authors showed that different lightweight libraries and algorithms can be deployed on edge nodes such as smartphones and enable real-time data analytics.

2.2. Data reduction in IoT applications

Different data reduction schemes for energy saving in IoT applications have been proposed in the state of the art. In [7][8], the authors proposed data reduction approaches based on adaptive sampling. These approaches work by studying the level of variance between the collected data over a certain time frame, and dynamically adjusting the sampling frequency of the devices. Adaptive sampling approaches work well in applications where the collected time series are stationary. In the case of quickly varying data, these approaches perform poorly. In [9], the authors proposed a data reduction mechanism based on dual prediction. The proposed mechanism works by building a model that describes the sensed phenomenon and deploying it on both the edge node and the IoT devices. The advantage of prediction approaches is that the model at the edge predicts the sensed measurement without requiring any radio communication unless the prediction error exceeds a predefined threshold. However, such prediction mechanisms suffer when it comes to devices like a high frequency motion sensor, where the data collection frequency is high and the data varies quickly.

Numerous aggregation and compression approaches that take advantage of the temporal correlation in the collected data have been proposed [10, 11, 12, 13, 14]. In [10], the authors proposed a data filtering technique based on the Pearson coefficient metric. This method works by recursively dividing the dataset into two equal parts and aggregating the data based on the correlation between the subsets. In [11], the author proposed a data aggregation method for incoming data stream in IoT based monitoring systems. The proposed method is an approximation with 'extremums' technique that reduces the volume of data to be stored or transmitted. The results show that this method was able to achieve a compression of up to 10 times on temperature data. In [12, 13], the authors proposed data compression techniques that take advantage of the temporal correlation in the collected data. The proposed techniques are based on the simple and computationally efficient 1-D Discrete Wavelet Transform (DWT) via lifting scheme and the Differential Pulse Code Modulation (DPCM). The aforementioned data reduction techniques, in addition to many others proposed in the literature, perform well on stationary univariate time series. However, an important number of IoT devices nowadays include more than one sensor and are able to collect multiple features. Therefore, data reduction techniques that work efficiently on multivariate time series are required.

Compressive Sensing (CS) and transform domain compression techniques that are usually used for images have been proposed as well for multivariate time series compression in IoT applications. In [15], the authors proposed a multisignal compression technique based on the theory of fuzzy transform. The proposed method has been applied on multisignal environmental data collected by a wireless sensor network and reduced the data by approximately two times. In [16], the authors proposed the 2-D lifing wavelet transformation to compress multisignal data collected from different sensor nodes. The proposed method uses the Haar wavelet and achieves a compression ratio of 1.33 and recovery accuracy of 98.4%. Transform domain compression techniques are characterized by the ability to recover the data accurately. However, the compressing performance of these techniques remains limited. On the other hand, CS theory has emerged as an efficient approach for energy-efficiency in IoT applications in recent years [17] [18]. By taking advantage of the signal sparsity, CS approaches assure an accurate signal recovery by sampling signals at a much lower rate than the traditional Shannon-Nyquist theorem. Nevertheless, CS techniques suffer when dealing with non-sparse multi-dimensional signals containing diverse features with different scales of values.

The major drawbacks of the above-mentioned propositions is that they yield low compression ratio on non-stationary multi-sensor data and they are not tested on real devices. This paper proposes a data compression technique for IoT applications and resource constrained devices that works efficiently on multivariate time series and implemented on a real wearable device. In the following sections, the proposed lossy compressor is presented and the impact of information loss on data analytics at the edge is studied.

3. Error-bounded lossy compression

In this paper, a lighweight version of the work presented in [19] is given. The authors in [19] proposed a fast error-bounded lossy compression scheme namely SZ for High Performance Computing (HPC) applications. This compression scheme has been proposed to deal with the huge amounts of data generated during the execution of HPC applications. The original SZ compresses input data files that are in binary format and can have different data shapes and data types (single-precision and double-precision). In this work, we propose to adapt the SZ algorithm for IoT devices by considering only the floating point data type and discarding the other types which make the code smaller in size and easier to compile on tiny devices. Moreover, the algorithm was adapted to take a 1-D array of float sensor data as input and return a byte array that is going to be transmitted to the edge node. The motivations behind choosing SZ for IoT applications are as follows:

- SZ allows the compression of multivariate time series containing diverse features with different scales
- SZ allows the control of information loss by using an error bound
- SZ leads to higher compression ratio than the multi-dimensional transform domain compression techniques

The proposed compression scheme is defined in Algorithm 1. It is considered that the data are transmitted to the edge after each period P of time t. The collected data are in the form of $M \times N$ array, where M denotes the number of readings and N denotes the number of features. For example, consider a motion sensor that collected 128 gyroscope and accelerometer readings for the three coordinate axes after a period P. In that case, M is equal to 128, and N to 6.

To begin, the 2-D array is converted to the 1-D array (Algorithm 1, line 4). Then, the flattened array is compressed using the lossy SZ technique. Finally, the resulted binary array is transmitted to the edge. Algorithm 2 presents the main steps employed by the adapted SZ compression scheme. Note that the adaptation has been done by extracting the necessary functionalities from the original SZ to make it fit on wearables and resource constrained devices.

Algorithm 1 Proposed compression scheme

```
Require: $E$ (error bound)
1: while $Energy > 0$ AND $Sensors\_status = ON$ do
2:   for each period do
3:     $data[M, N] \leftarrow$ collected sensors data
4:     $input[M \times N] \leftarrow Flatten(data)$
5:     $bin\_output \leftarrow adapted\_SZ(input, E, M, N)$ (Alg 2)
6:     $transmit\_Data(bin\_output)$
7:   end for
8: end while
```

The SZ compressor starts by compressing the 1-D array using adaptive curve-fitting models. The bestfit step employs three prediction models: Preceding Neighbor Fitting (PNF), Linear-Curve Fitting (LCF), and Quadratic-Curve Fitting (QCF). The difference between the three models resides in the number of precursor data points required to fit the original value. The adopted model is the one that yields the closest approximation. Note that the fitted data are transformed into integer quantization factors and encoded using Huffman tree. In the case when none of the prediction models in the curve-fitting step satisfies the error bound, the data point is marked as unpredictable and is then encoded by analyzing the IEEE 754 binary representation. (Algorithm 2, line 2).

Algorithm 2 Adapted SZ steps

```
Require: input (1-D array), $E$ (error bound),$M$ (num rows),$N$ (num columns)
Ensure: output (binary array)
1: Bestfit Curve-Fitting Compression
2: Compressing Unpredictable Data
```

As for the error bound, the absolute error bound has been used in which the compression/decompression errors are limited to be within an absolute error. For instance, if the value of a data point is considered to be X, an absolute error bound of 10^{-1} means that the decompressed value should be in the range $[X - 10^{-1}, X + 10^{-1}]$.

4. Case Study

The combination of IoT, cloud computing, and healthcare has taken a lot of attention during the past years. Among the major challenges that face the healthcare applications are:

- Latency due to communication between cloud and IoT devices
- Limited network bandwidth due to the high amount of generated data
- High cost of privacy and security breaches

Here is where the benefits of edge computing take place. As mentioned in previous sections, edge computing can be considered a major solution for latency and bandwidth challenges in IoT and healthcare applications. In a similar way, efficient on-node data compression can increase the lifetime of the application, and reduce the size and the number of transmitted packets which may affect the latency and network bandwidth positively. However, the integrity of the data and loss of information due to compression remain critical issues for healthcare applications. In this section, the impact of data compression and energy efficiency on the analysis of medical data at the edge is studied. Let us consider the case of driving behaviour monitoring and stress detection where the physiological signals are continuously collected from a driver and transmitted to the edge node for analysis and detection of stress level.

In the following, the dataset used in our work is described, then the processing and analysis of the data are discussed, and finally, the model used for stress level detection is presented.

4.1. Dataset

The Stress Recognition in Automobile Drivers database published on PhysioNet has been used in this work [20][21]. This dataset contains multiple physiological signals recorded from healthy volunteers, taken while they were driving on a specified route in and around Boston, Massachusetts. The driving task done by each driver varies from about 50 min to 1.5 h and can be divided into six sections as shown in Figure 2:

- Rest [1,6]: The resting periods in the beginning of the driving task and at the end of it are labeled as ”low stress”

- City [2,5]: Driving in the city periods in sections 2 and 5 are labeled as ”high stress” since the subjects drove in a busy main street and frequently handled the traffic conditions and the unexpected emergencies created by cyclists and jaywalkers

- Highway [3,4]: Driving on the highway periods in sections 3 and 4 are labeled as ”moderate stress”

Note that the labeling of the data has been validated by drivers’ self-reported questionnaires in [20].

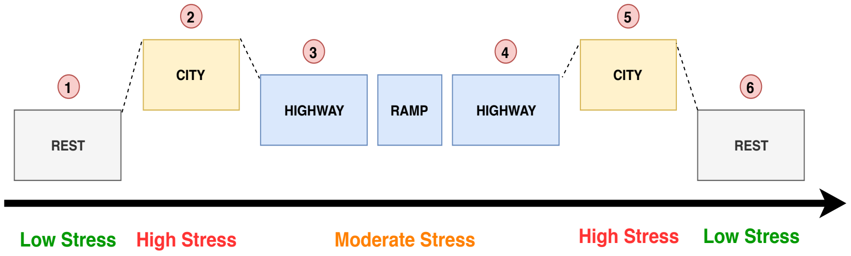

Figure 2: Driving task

This study is conducted on 9 drivers among 17 available in the database since the marker indicating the driving sections (rest, city, highway) is not present in all drivers files. Five physiological signals were used for stress detection, namely electrocardiogram (ECG), heart rate (HR), galvanic skin response (GSR) of the hand and foot, and respiration rate (RR).

4.2. Data processing and analysis

For each driver data file, the noise has been removed from HR and GSR signals following [22]. In order to study the impact of data compression on the information contained in the data, a feature extraction phase is used from which time-domain and frequency-domain features are extracted from the ECG signal in addition to the two main components of the GSR signal namely: Skin Conductance Level (SCL), and Skin Conductance Response (SCR). The objective of this step is to compare the features extracted from the original data with the features extracted from the compressed data.

Table 1 describes the features extracted from the ECG signal by applying time and frequency domain analysis such as FIR-filters and fast fourier transform methods. In order to analyze the GSR signal, which can be considered

Table 1: Features extracted from ECG signal

Feature	Description
RMSSD	Root mean square of the Inter-beat (RR) Intervals (the time intervals between consecutive heart beats)
meanNN	Mean RR interval
sdNN	Standard deviation RR interval
cvNN	Coefficient of Variation (CV), i.e. the ratio of sdNN divided by meanNN
CVSD	Coefficient of variation of successive differences, i.e. the RMSSD divided by meanNN
medianNN	Median of the absolute values of the successive differences between the RR intervals
madNN	Median Absolute Deviation (MAD) of the RR intervals
mcvNN	Median-based Coefficient of Variation (MCV), i.e. the ratio of madNN divided by medianNN
pNN20	The number of interval differences of successive RR intervals greater than 20 ms divided by the total number of RR intervals
pNN50	The number of interval differences of successive RR intervals greater than 50 ms divided by the total number of RR intervals

an important sensitive measure for emotional arousal, we extract the slow variation (SCL) and the faster alterations (SCR) following a convex optimization approach proposed in [23]. Table 2 shows the features extracted from the GSR signal.

4.3. Stress detection using Feed-Forward Neural Network (FFNN)

This section considers the features shown in Tables 1 and 2 in addition to heart rate (HR) and respiration rate (RR) as a single sequence in which a label is assigned. The prediction task is a supervised sequence classification task. A

Table 2: Features extracted from GSR signal

Feature	**Description**
meanGSR	Mean value of GSR signal
meanSCL	Mean value of SCL
slopeSCL	Difference between max and min values of SCL
meanSCR	Mean value of SCR
maxSCR	Max value of SCR

FFNN is a network that contains a large number of neurons, arranged in layers: one input layer, one or more hidden layers, and one output layer.

Figure 3 shows the FFNN architecture used in the classification task. The neural network consists of 17 input neurons, corresponding to a sequence of 17 physiological features. Additionally, the neural network is provided with a correct label of that sequence. That is, whether the sequence corresponds to low stress, moderate stress, or high stress. Providing that edge analytics require lightweight algorithms to perform reasonable machine learning [6], we have used a network with a minimized size that maintains accuracy. The implemented network has four hidden layers consisting of 60 neurons, which use the ReLU activation function [24]. The number of layers The network was trained with a variant of stochastic gradient descent 'Adam' [25] and categorical cross entropy. Furthermore, Dropout regularization technique [26] has been used to prevent overfitting and 10-Fold Cross Validation has been used to validate and find the optimal set of hyperparameters for the model. Finally, the network has been trained for 300 epochs with a learning rate of 0.01.

5. Experimental Results and Analysis

In the following sections, the results of applying the aforementioned data compression technique on physiological data are presented. Two metrics are discussed in the following: data reduction, and the impact of information loss on the prediction accuracy.

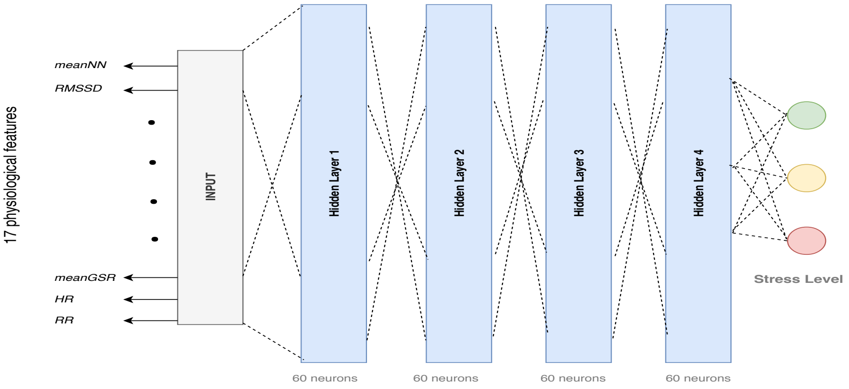

Figure 3: FFNN

5.1. Data reduction and energy conservation

The data sets and the signals were recorded from the drivers at 496 Hz. In order to test the efficiency of the proposed compression scheme, we have deployed the data on the SD card of a Polar M600 wearable. In the same way, the compression technique written in C language using Android NDK toolset was implemented. After each period $p = 1\,min$, an array $data[M, N]$ is transmitted using Android (Bluetooth Low Energy) BLE to the edge device, which is a local PC where the received data are processed and analyzed. The results are then transferred to the cloud. Note that M, the number of readings, is equal to 148800 and N, the number of features, is equal to 5.

The experimentation has been conducted for around 4 hours, equivalent to 241 periods. Figure 4 shows the difference between the amount of data transmitted with and without compression. The x-axis denotes the periods, and the y-axis denotes the number of bytes needed to represent the data in logarithmic scale. The transmission of the original data after each periods requires around 2976000 bytes, while the number of bytes required for transmitting the compressed data varies between 28732 bytes and 41602 bytes. Thus, reducing the transmitted data by up to 103 times.

Figure 5 describes the change of the wearable battery level over 241 periods for five different scenarios:

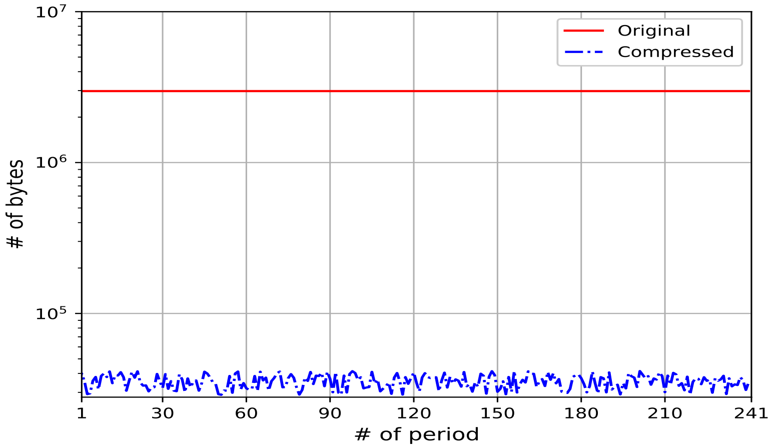

Figure 4: Amount of original and compressed data transmitted over 241 periods

- Black line: the wearable device is in the idle state
- Red line: the wearable device is continuously collecting data (motion and heart rate sensors are turned on)
- Yellow line: the wearable device is collecting data and running the compression algorithm after each period
- Green line: the wearable device is collecting data, and performing compression and transmission after each period
- Blue line: the wearable device is collecting data, and performing data transmission after each period (no compression)

The results clearly show the impact of data reduction on the communication task energy consumption. The battery level of the device continuously collecting data was decreased to around 86% after 241 periods. Note that by comparing the sensing and computation tasks with the idle state, it can be noticed that these tasks contribute a little in the energy consumption of the device. On the other hand, when applying the proposed data compression scheme prior to transmission, the battery level was decreased to 83% while sending the data

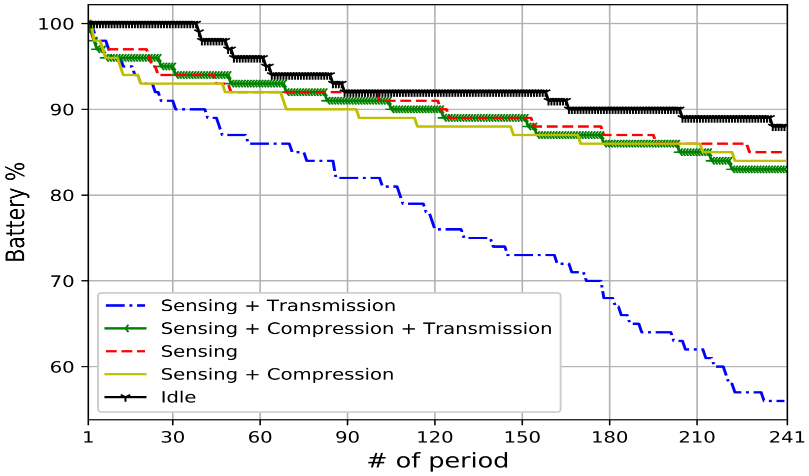

Figure 5: Polar M600 battery level over 241 periods

without compression decreased the battery level to around 56%. As a result, the device lifetime could be increased by up to 27% after 4 hours.

5.2. Loss of information and stress detection

In this section, the impact of information loss on the prediction accuracy of the proposed FFNN for stress level detection is studied. For each of the drivers datasets, a sliding window of 30 seconds with 75% overlap on the data is applied, and the physiological features described in the previous section are extracted from each window. Note that for each window, the extracted features form a set/sequence labeled as low, moderate, or high stress that is going to be fed as an input to the neural network.

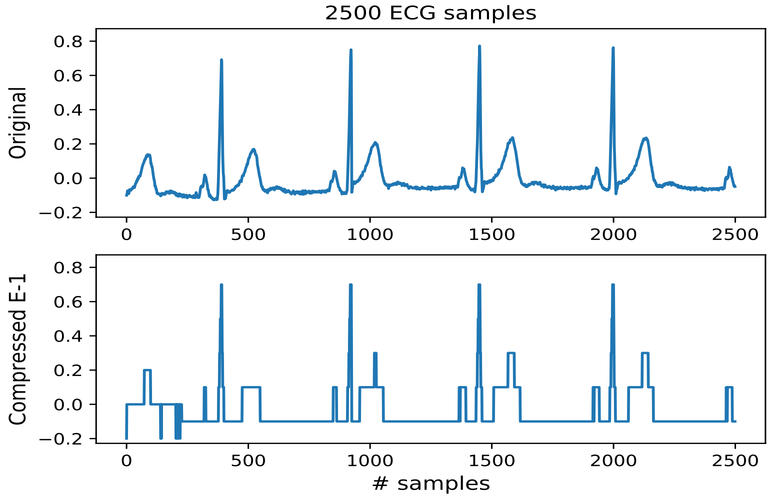

Figure 6: 2500 original ECG samples vs compressed ECG samples

Figure 6 compares the original ECG signal transmitted to the edge without compression with the compressed signal using an error bound of 10^{-1}. Even though the compression has affected slightly the shape of the signal, the positions of the peaks that are used to exploit the signal and extract the most important features remain unchanged. Note that one of the advantages of the SZ algorithm is that the error bound is controllable and can be initialized according to the medical need. In other words, the trade off between compression ratio and loss of information can be easily controlled by the medical staff.

Table 3 shows the Root Mean Square Error (RMSE) between the features extracted from the compressed and original ECG and GSR signals for the 9 drivers. Notice that the average RMSE for each feature is small, and the important features such as heart rate, respiration rate, and R-R interval have a RMSE close to zero, which means that the compression had very low impact on the information loss.

In order to fully answer the stated problem in section 1, the sequences of features are randomly divided into train and test sets (75%/25%). Two models are considered, the first one was trained on the sequences of features extracted

	Driver #									***Average***
	4	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**16**	
HR	0.27	0.08	0.07	0.10	0.04	0.08	0.07	0.07	0.26	**0.11**
RMSSD	1.57	1.68	1.18	0.94	1.20	2.45	3.77	2.28	2.42	**1.94**
meanNN	0.91	1.24	0.24	0.23	0.25	1.02	0.30	0.70	1.09	**0.66**
sdNN	0.61	1.01	0.29	0.33	0.58	1.12	1.65	0.87	1.44	**0.87**
cvNN	0	0	0	0	0	0	0	0	0	**0**
CVSD	0	0	0	0	0	0	0	0	0	**0**
medianNN	2.19	1.4	0.67	1.34	1.01	2.08	1.74	1.06	1.63	**1.45**
madNN	1.91	1.19	1.01	1.22	1.12	1.96	1.95	0.88	1.52	**1.41**
mcvNN	0	0	0	0	0	0	0	0	0	**0**
pNN50	1.78	2.2	1.23	1.51	1.91	1.92	2.24	1.56	1.27	**1.72**
pNN20	2.52	3.63	1.70	1.95	1.93	2.39	1.80	1.92	3.38	**2.35**
mean_gsr	0	0	0.01	0.02	0.02	0.01	0.02	0	0	**0**
mean_scl	0.23	0.22	0.23	0.21	0.21	0.20	0.20	0.19	0.23	**0.21**
slope_scl	0.38	0.36	0.34	0.34	0.28	0.33	0.34	0.29	0.44	**0.34**
mean_scr	0.22	0.22	0.23	0.21	0.21	0.20	0.20	0.19	0.23	**0.21**
max_scr	0.37	0.31	0.30	0.31	0.23	0.30	0.23	0.27	0.41	**0.30**
RR	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	**0.02**

Table 3: Root mean square error between the features extracted from original data and the features extracted from reconstructed data

from the original data and the second one on the sequences of features extracted from the compressed data. For hyperparameter optimization, 10-fold cross validation is performed on the training sets, and then our models are evaluated on the test sets. Table 4 summarizes the stress level detection performance of the aforementioned models. The results show that the average accuracy achieved by the two models is 98%. It can be noticed that not only the compression did not affect the prediction accuracy but even improved it in some cases such as for driver 7 and driver 12, which is due to the denoising capability of the compression technique.

	Accuracy	
	Original	***Compressed***
Driver 4	1.0	0.99
Driver 6	1.0	1.0
Driver 7	0.97	0.98
Driver 8	0.98	0.98
Driver 9	0.98	0.97
Driver 10	0.99	0.99
Driver 11	0.92	0.94
Driver 12	0.99	0.99
Driver 16	0.99	0.98
Average	**0.98**	**0.98**

Table 4: FFNN prediction accuracy on test sets (25%) corresponding to sequences of features extracted from original and compressed data respectively

6. Discussion

Most of the IoT devices nowadays are equipped with multiple sensors and are able to collect different types of data. Thus, the compression techniques must deal with multisensor readings at a single device. Furthermore, in the case of real-time or near real-time applications, the reconstruction (decompression) time of the algorithm must be small in order to pass the data to the machine learning model and return the prediction/classification results to the user in the shortest time possible. Although transform-based compression and compressed sensing (CS) can be used for dealing with multisensor readings, these methods have several disadvantages that make the choice of SZ for this type of problems is the best solution. Transform-based compression techniques transform raw data to a set of coefficients, and need to be followed by an entropy coding step to encode the coefficients in order to achieve an acceptable reduction rate. On the other hand, CS requires that a signal is sparse in some domain and doesn't contain noise in order to achieve an "exact" reconstruction, which is not the case in real world applications. CS is also an asymmetric algorithm, which means that the decompression needs a higher computational complexity than that of the compression and longer time in order to recover the data, making it inefficient for applications that require fast responses. Another issue for transform-based techniques and CS is adaptability. In other words, the compression algorithm must adapt and perform well across different applications, subjects, and activities. For instance, the multivariate time series used in this paper contains different variables having different statistical characteristics. So in order to apply CS to multivariate data, each of the variables contained in the data must meet the conditions needed by CS to work perfectly, which is not always the case.

Data size (bytes)	Compression time (seconds)	Decompression time (seconds)
11904000	0.036	0.05
2976000	0.012	0.01
992000	0.005	0.006
99200	0.003	0.002

Table 5: Average compression/decompression time for the SZ algorithm on different data sizes

Table 5 shows the time needed by the proposed SZ algorithm deployed on a wearable device (Polar M600) to compress and decompress different sizes of input multivariate data. It can be seen that the average compression/decompression time is small, thus making SZ suitable for near real-time applications. For the above-mentioned reasons, the proposed SZ can be a better candidate for multisensor readings compression not only for its fast compression/decompression and high compression rate, but also for its ability to adapt for different applications and scenarios.

7. Conclusion

Since bringing intelligence closer to IoT devices reduces network latency and energy consumption due to radio communications, data reduction can be seen as an additional solution to increase the lifetime of IoT devices. In this paper, an energy efficient data reduction scheme for IoT-Edge applications was proposed. The proposed scheme is based on an error-bounded lossy compressor designed for high performance computing applications that produce large amounts of data during the execution. The compression algorithm was adapted to fit on a Polar M600 wearable and its performance was tested on medical multivariate time series. The results showed that the we were able to reduce the amount of transmitted data to the edge device by up to 103 times and thus to increase the lifetime of the wearable. Furthermore, we considered the use case of drivers stress recognition and studied the impact of lossy data compression on the analysis, exploit, and classification of medical data. The results showed that the information extracted from compressed data was valid, and the classification accuracy obtained from training the model on features extracted from compressed data did not decrease.

Acknowledgment

This work is partially funded by the EIPHI Graduate School (contract "ANR-17-EURE-0002")

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet of Things Journal 3 (5) (2016) 637–646. doi:10.1109/JIOT.2016.2579198.

[2] G. Anastasi, M. Conti, M. D. Francesco, A. Passarella, Energy conservation in wireless sensor networks: A survey, Ad Hoc Networks 7 (2009) 537 – 568.

[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. doi:10.1145/2528948.
URL http://doi.acm.org/10.1145/2528948

[4] A. P. Miettinen, J. K. Nurminen, Energy efficiency of mobile clients in cloud computing, in: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud'10, USENIX Association, Berkeley, CA, USA, 2010, pp. 4–4.
URL `http://dl.acm.org/citation.cfm?id=1863103.1863107`

[5] H. Li, K. Ota, M. Dong, Learning iot in edge: Deep learning for the internet of things with edge computing, IEEE Network 32 (1) (2018) 96–101. `doi:10.1109/MNET.2018.1700202`.

[6] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, D. S. Nikolopoulos, Challenges and opportunities in edge computing, in: 2016 IEEE International Conference on Smart Cloud (SmartCloud), 2016, pp. 20–26. `doi:10.1109/SmartCloud.2016.18`.

[7] D. Laiymani, A. Makhoul, Adaptive data collection approach for periodic sensor networks, in: 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC), IEEE, 2013.

[8] C. Habib, A. Makhoul, R. Darazi, R. Couturier, Real-time sampling rate adaptation based on continuous risk level evaluation in wireless body sensor networks, in: 2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), IEEE, 2017.

[9] G. B. Tayeh, A. Makhoul, D. Laiymani, J. Demerjian, A distributed real-time data prediction and adaptive sensing approach for wireless sensor networks, Pervasive and Mobile Computing 49 (2018) 62 – 75.

[10] H. Harb, A. Makhoul, C. A. Jaoude, En-route data filtering technique for maximizing wireless sensor network lifetime, in: 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC), 2018, pp. 298–303. `doi:10.1109/IWCMC.2018.8450348`.

[11] V. Alieksieiev, One approach of approximation for incoming data stream in iot based monitoring system, in: 2018 IEEE Second International Conference on Data Stream Mining Processing (DSMP), 2018, pp. 94–97. doi:10.1109/DSMP.2018.8478466.

[12] J. Azar, A. Makhoul, R. Darazi, J. Demerjian, R. Couturier, On the performance of resource-aware compression techniques for vital signs data in wireless body sensor networks, in: 2018 IEEE Middle East and North Africa Communications Conference (MENACOMM), 2018, pp. 1–6. doi:10.1109/MENACOMM.2018.8371032.

[13] 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.

[14] H. Harb, A. Makhoul, C. A. Jaoude, A real-time massive data processing technique for densely distributed sensor networks, IEEE Access 6 (2018) 56551–56561.

[15] M. Gaeta, V. Loia, S. Tomasiello, Multisignal 1-d compression by f-transform for wireless sensor networks applications, Appl. Soft Comput. 30 (C) (2015) 329–340. doi:10.1016/j.asoc.2014.11.061.
URL http://dx.doi.org/10.1016/j.asoc.2014.11.061

[16] L. Cheng, S. Guo, Y. Wang, Y. Yang, Lifting wavelet compression based data aggregation in big data wireless sensor networks, in: 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS), 2016, pp. 561–568. doi:10.1109/ICPADS.2016.0080.

[17] A. Deligiannakis, Y. Kotidis, N. Roussopoulos, Compressing historical information in sensor networks, in: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, SIGMOD '04, ACM,

New York, NY, USA, 2004, pp. 527–538. doi:10.1145/1007568.1007628. URL http://doi.acm.org/10.1145/1007568.1007628

[18] A. Fragkiadakis, P. Charalampidis, E. Tragos, Adaptive compressive sensing for energy efficient smart objects in iot applications, in: 2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace Electronic Systems (VITAE), 2014, pp. 1–5. doi:10.1109/VITAE.2014.6934488.

[19] 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

[20] J. A. Healey, R. W. Picard, Detecting stress during real-world driving tasks using physiological sensors, IEEE Transactions on Intelligent Transportation Systems 6 (2) (2005) 156–166. doi:10.1109/TITS.2005.848368.

[21] A. L. G. *et al.*, Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals, Circulation.

[22] S. Ollander, Wearable sensor data fusion for human stress estimation, Master's thesis, Technical University of Linkoping University (2015).

[23] A. Greco, G. Valenza, A. Lanata, E. P. Scilingo, L. Citi, cvxeda: A convex optimization approach to electrodermal activity processing, IEEE Transactions on Biomedical Engineering 63 (4) (2016) 797–804. doi:10.1109/TBME.2015.2474131.

[24] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML'10, Omnipress, USA, 2010, pp. 807–814.
URL http://dl.acm.org/citation.cfm?id=3104322.3104425

[25] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, CoRR abs/1412.6980. arXiv:1412.6980.
URL http://arxiv.org/abs/1412.6980

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, Journal of Machine Learning Research 15 (2014) 1929–1958.
URL http://jmlr.org/papers/v15/srivastava14a.html