An Efficient Lossless Compression Algorithm for Electrocardiogram Signals

Giuseppe Campobello[1], Antonino Segreto[1], Sarah Zanafi[2], Salvatore Serrano[1]
[1]Department of Engineering - University of Messina (Italy)
[2]Faculty of Science Tetouan, University Abdelmalek Essaadi (Morocco)

***Abstract*—This paper focuses on a novel lossless compression algorithm which can be efficiently used for compression of electrocardiogram (ECG) signals. The proposed algorithm has low memory requirements and relies on a simple and efficient encoding scheme which can be implemented with elementary counting operations. Thus it can be easily implemented even in resource constrained microcontrollers as those commonly used in several low-cost ECG monitoring systems. Despite its simplicity, simulation results carried out on real-world ECG signals show that the proposed algorithm achieves higher compression ratios as even compared to other more complex state-of-the-art solutions.**

I. Introduction

Nowadays there are an increasing number of people with chronic cardiovascular conditions and related diseases [1]. As a consequence an escalating level of supervision and medical management systems is necessary.

Wireless Sensor Networks (WSNs) and, more recently, Internet of Things (IoT) platforms have been proposed as a solution to this problem and are considered the key enablers for future health monitoring systems [2] [3]. Usually it is assumed that this kind of devices have enough processing and storage resources to run powerful and complex algorithms [4] and to store hundreds of megabytes of data [5]. However, in practice, resource constrained microcontrollers with a few kilobytes of memory are used in IoT platforms and WSNs [6] and are preferred with the aim to achieve low cost monitoring systems [7] [8]. In this context, compression techniques can be used with the aim of reducing storage resources and increasing lifetime of battery-powered devices [9].

Compression algorithms can be broadly classified in lossy and lossless compression algorithms [10], [11]. In both cases the first metric considered to evaluate their performance is the Compression Ratio (CR), here defined as the ratio between the number of bits before and after compression. As general rule, lossy compression algorithms allow to achieve much higher compression ratios, however lossless algorithms are preferred in several biomedical applications to ensure that waveform details are not lost causing errors in medical diagnosis [12]. Moreover, in most countries, medical regulatory standards do not endorse lossy compression techniques [13] [14] [15].

Although several lossless compression algorithms exist, most of them are not suitable when only limited storage and computational resources are available [16] [17]. Therefore, in the case of resource constrained devices, the tradeoff between computational complexity and the achievable compression ratio must be further considered.

In this paper we present a novel lossless compression algorithm based on a simple and efficient encoding scheme and which can be easily implemented even in resource constrained microcontrollers.

The proposed compression algorithm has been tested with real-world electrocardiogram (ECG) signals obtained by the MIT-BIH Arrhythmia Database [18] which contains 48 two-channel ambulatory ECG recordings digitized at 360 samples-per-second with 11-bit resolution.

Simulation results reported in this paper show that the proposed algorithm achieves higher compression ratios as even compared to state-of-the-art lossless compression algorithms based on more complex predictors and encoding schemes.

The rest of this paper is organized as follows: in Sec. II related works are discussed; in Sec. III the proposed algorithm is introduced; in Sec. IV performance and complexity of the proposed algorithm and other state-of-the-art lossless compression schemes are compared. Finally, conclusions and future works are drawn in Sec. V.

II. Related works

The basic idea behind several ECG lossless compression algorithms is to exploit temporal correlation to reduce redundancy. Accordingly, many compression algorithms operate in two phases:

- in the first phase, a predictor is used; basically, in this phase a few previous samples are used to obtain an accurate estimate $\hat{x}_i$ of the actual sample x_i. The most simple predictor is the zero order predictor, i.e. $\hat{x}_i = x_{i-1}$, but several other techniques exist ranging from higher order predictors and interpolators [19] [20] to more computationally intensive solutions based on neural networks [21] or fuzzy logic [22].
- In the second phase, the difference $r_i = x_i - \hat{x}_i$, henceforward named residue, is encoded using a codebook specifically optimized to achieve a mean codeword length near the Shannon's entropy [23]. Variable-length codes (VLC) such as Huffman codes [24] or Golomb codes [25] can be used in this phase.

Several compression algorithms that exploit the above approach have been proposed in literature, some of which are reviewed below.

ISBN 978-90-827970-1-5 © EURASIP 2018

In [26] the authors proposed a lossless ECG compressor based on an adaptive predictor and a two-stage Huffman encoder. Basically, the adaptive scheme selects the best predictor among a first-order and a second-order linear predictor; residues are then encoded with a two-steps Huffman encoder implemented using two look-up tables (LUTs). Implementing the algorithm in microcontrollers is feasible and in the case of the MIT-BIH Dataset the achieved average compression ratio is 2.43. However, as observed in [27], the authors considered that residues obtained from 11-bit samples can be always represented with 9 bits; instead, at least 13 bits must be used for a true lossless implementation based on a second-order predictor.

In [27] authors use a first-order predictor and a dynamic data packaging scheme. The packaging scheme basically removes unnecessary zeros and select the best frame format that can accommodate a set of residues. The proposed solution is simple and its implementation on microcontrollers is straightforward; however, in comparison to [26], the achieved compression ratio is lower (2.25).

In [28] an adaptive predictor based on a fuzzy decision controller has been proposed. A set of three fuzzy rules has been used to select the best predictor among a set of height different linear predictors. Then a two-stage entropy encoder similar to that proposed in [26] is used to encode the residues. In comparison to the above mentioned solutions, the authors obtained a higher compression ratio (i.e. 2.53); however it should be considered that the entropy encoder and the set of fuzzy rules have been specifically tuned with the same dataset used to evaluate the compression ratio.

In [29] authors proposed a solution based on an adaptive predictor and a variable length encoding scheme. Basically, residues within the range $-7 < r_i \leq 8$ are encoded with 4 bits each and, in the case of larger residues, 12 bits are used to directly output the input data. The authors stated that, in the case of the MIT-BIH dataset, the proposed scheme is able to obtain an average compression ratio of 2.67. However, the adaptive predictor is based on a threshold value which has been specifically tuned for the MIT-BIH dataset. Moreover, the above compression ratio has been evaluated with an incorrect resolution of 12 bits-per-sample. Considering that compression ratios increase with the resolution of input data and that actual resolution of the MIT-BIH samples is 11 bits [18], their effective compression ratio is $2.67 \cdot \frac{11}{12} \approx 2.45$. We argue that this mistake was due to the fact that MIT-BIH recordings are stored in different formats; the most common is the 212 format where, for sake of efficiency, 11-bit samples are stored as 2 samples per 3 bytes (i.e. 12 bits per sample).

In [30] authors improved the algorithm proposed in [27] by combining a fourth order adaptive predictor, a Huffman code and a modified dynamic data packaging scheme. In the case of the MIT-BIH Dataset they achieved an average compression ratio of 2.38.

In [22] authors proposed a novel prediction method based on fuzzy set decision, a particle swarm optimiser (PSO) and a six-region Huffman encoder. In the case of the MIT-BIH Dataset the achieved average compression ratio is 2.84. However, this impressive compression ratio has been obtained with a complex solution which relies on an off-line optimization procedure. In particular, 64 threshold values have been specifically tuned for the MIT-BIH dataset using a training scheme based on PSO that is too complex to be implemented in low-cost microcontrollers.

In [31] authors proposed a solution based on an adaptive linear predictor and a small single stage Huffman encoder. In the proposed solution, infrequent high value residues are divided in two parts which are encoded in two clock cycles using the same Huffman dictionary. The authors stated that in comparison to [26] a 2.1X preformance improvement can be achieved. However, the proposed solution is far from being flawless. Firstly, their implementation assumes that residues can be always expressed with two decimal digits (in general this constrain is not verified and it is worth noting that the authors tested their implementation only on ten signals of the MIT-BIH dataset). Secondly, the 2.1X improvement has been achieved without considering that some bits are necessary to encode also the type of the used predictor; when this precode is considered the actual compression ratio is less than 2.2 (as can be observed from Fig.6 of their paper).

Considering the aforementioned works it is possibile to state that the design of a low complex lossless compression scheme with a compression ratio higher than 2.67 is a challenging task. In particular, at the best of our knowledge, a compression ratio larger than 2.67 has been obtained only with complex predictors that cannot be easily implemented with low cost microcontrollers.

In the next Sections we present a new lossless compression algorithm ables to achieve the above compression ratio using a simple encoding scheme and a zero order predictor.

III. Proposed algorithm

The proposed algorithm is based on an efficient encoding scheme for binary sequences recently proposed in [32] and named RAKE. In [32] the authors showed that in the case of sparse sequences, i.e. sequence with a few non-zero bits, the RAKE algorithm is able to outperform other encoding techniques such as the Run-Length Encoding (RLE) [11].

For the sake of readability, we report a short description of the RAKE algorithm in the next subsection.

A. RAKE algorithm

The RAKE algorithm is able to encode positions of non-zero bits in an efficient manner. We can explain the RAKE algorithm by considering a sliding window of length T that moves forward over the original (uncompressed) binary sequence (see Figure 1). The window catches T bits at a time and an output codeword is generated accordingly to the following two possible cases:

1) There are no set bits within the window (i.e. all T bits are zeros). In this case a single zero bit codeword is used and the sliding window is moved forward by T positions;

ISBN 978-90-827970-1-5 © EURASIP 2018

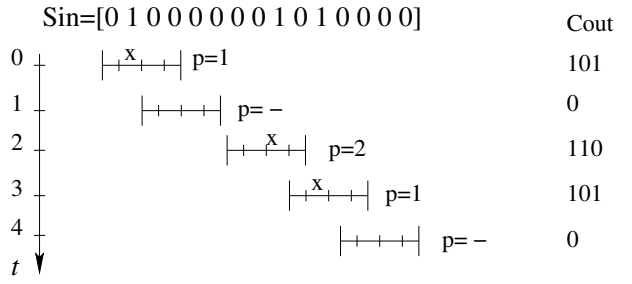

Fig. 1. Example of RAKE compression algorithm ($T = 4$).

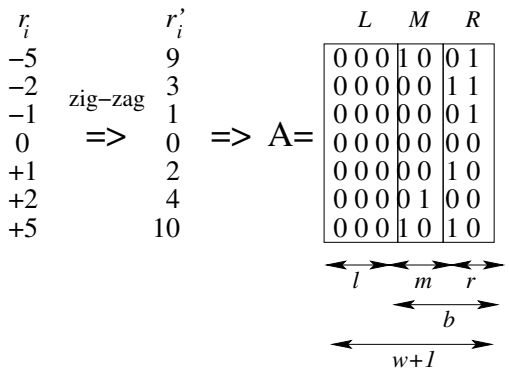

Fig. 2. Zig-zag encoded residues and related matrix $A = [L, M, R]$

2) At least a set bit is found. In this case a codeword of $1 + \lceil \log_2 T \rceil$ bits is generated where the first bit is set to 1 and the other $\lceil \log_2 T \rceil$ bits are used to encode the position $p \in [0, ..., T-1]$ of the first non-zero bit within the window. Then the window moves forward by $p + 1$ positions (i.e. immediately after the set bit that has been already encoded).

The above operations are repeated until the sliding window reaches the end of the sequence to be compressed. It is worth noting that only counting operations are needed for implementing the above procedure.

In Figure 1 a simple example is reported showing how the sequence $S_{in} = [010000001010000]$ of $n = 15$ bits is compressed by the RAKE algorithm to produce a compressed sequence $C_{out} = RAKE(S_{in}) = [10101101010]$ of 11 bits.

Accordingly to [32], the following expression provides the optimal value of T for a binary sequence of length n with k non-zero bits:

$$T = (\frac{n}{k} - 1) \cdot \ln(2) \tag{1}$$

Finally, note that the value of T is necessary to reconstruct the original sequence, thus a few bits must be added at the beginning of the compressed sequence. In [32] authors showed that T can be represented with only $\mathcal{O}(\log_2(\log_2(T)))$ bits by constraining the value of T to be a power of two. In this case the RAKE algorithm has a negligible overhead.

B. Using the RAKE algorithm for ECG signals

The RAKE algorithm can be used in combination with every transformation or pre-processing technique able to obtain a sparse binary sequence from the original data set. In particular, in [32] the authors proposed to represent residues using a dictionary based on a one-hot encoding scheme.

In this paper we apply the RAKE in a different manner by avoiding the use of dictionaries so that only the compressed and the incoming sequences have to be stored.

Henceforward we consider that ECG samples x_i are quantized and represented with w bits (i.e. $w = 11$ in the case of the MIT-BIH dataset) and that residues are obtained with a simple zero order predictor, i.e. $r_i = x_i - x_{i-1}$.

We introduce the following transformation, henceforward referred to as zig-zag encoding [33]:

$$r_i' = 2 \cdot |r_i| - (r_i < 0) \tag{2}$$

Zig-zag representation is similar to sign-magnitude representation but the sign bit of r_i is placed at the end of the binary string representing r_i' and, in the case of negative numbers (i.e. $r_i < 0$), the sign is used to reduce the magnitude of r_i'. It is possible to prove that the zig-zag transformation is invertible (i.e. r_i can be obtained from r_i').

We observe that at most $w+1$ bits are necessary to represent r_i'. As a consequence, a block of B zig-zag encoded residues, $\{r_i' : i \in [1, ..., B]\}$, can be represented as a matrix A of $B \times (w+1)$ bits. Here we assume that B and w are fixed and known values.

The basic idea of the proposed algorithm is to logically decompose the matrix A into three parts (i.e. three submatrices) and then apply different compression schemes to each submatrix. More precisely, as shown in Fig.2, the matrix A is decomposed as $A = [L, M, R]$ where:

- L is a $B \times l$ matrix, representing the Left part of A and composed by l all-zero columns;
- M is a $B \times m$ matrix, representing the Middle part of A and composed by m consecutive columns with at most $0.4 \cdot B$ non-zero bits for each column;
- R is a $B \times r$ matrix, representing the Right part of A and composed by the remaining r columns of A.

Let us observe that in some cases the above mentioned submatrix can be null (i.e. l, m and r can be zero) but their dimensions are always related by $l + m + r = w + 1$.

Note that the boundaries of the above matrices (and therefore the values of l, m and r) can be obtained by reading the matrix A column-by-column and counting the number of non-zero bits in each column. More precisely, let A_j be the j-th column of A and $nnz(A_j)$ a function able to count the number of non-zero bits in the column vector A_j, than it is possible to obtain l,m and r with the following procedure:

- initializing the variables $l = 0, m = 0, j = 1$
- while $((nnz(A_j) == 0)$ && $(j \leq w + 1))\{l$++; j++;$\}$
- while $((nnz(A_j) < 0.4B)$&&$(j \leq w + 1))\{m$++;j++;$\}$
- $r = w + 1 - l - m$

Finally note that no memory is necessary to store the above submatrices, in fact the above description is an alternative logical representation of the array A already used to store the encoded residues.

ISBN 978-90-827970-1-5 © EURASIP 2018

The above submatrices can be compressed as follows:

- L is an all-zeros matrix therefore dimensions of L are sufficient for its reconstuction; furthermore, we recall our assumption that B is a fixed and known value thus L can be compressed with $\lceil \log_2(w+2) \rceil$ bits, i.e. the number of bits necessary to represent l.
- M can be compressed using the RAKE algorithm; more precisely, RAKE is applied to each column of M by obtaining m compressed strings.
- R is not actually compressed but its elements are simply read as B words of r bits each, i.e. R_i with $i \in [1, ..., B]$.

Finally, compressed strings reprenting the above matrices are concatenated to obtain the compressed block

$$C_{out} = [l, m, RAKE(A_{l+1}), ..., RAKE(A_{l+m}), R_1, ..., R_B].$$

Note that from a computational point of view, we need only to obtain the values of l and m and then apply the RAKE algorithm to the columns of A belonging to M. Therefore only counting and simple arithmetic operations are needed for implementing the proposed algorithm.

The algorithm can be summarized as shown in Figure 3.

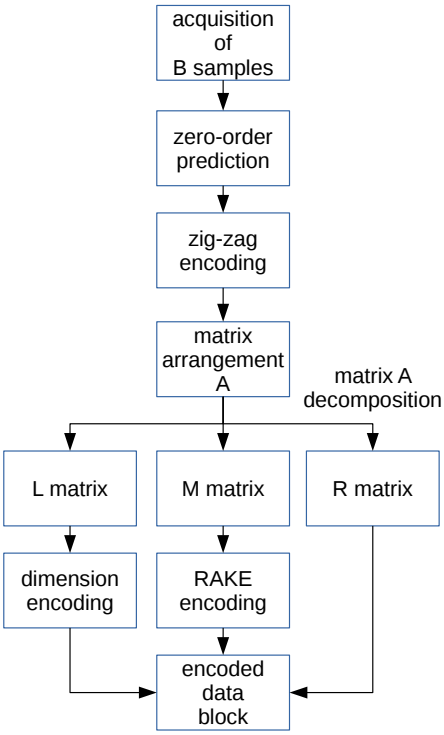

Fig. 3. Flowchart of the proposed algorithm

Finally, it is worth noting that to reconstruct the original ECG signal the first sample, i.e. x_0, is also necessary (this increases the overhead by w bits).

IV. Comparison results

In this section we compare the proposed compression algorithm with other state-of-the-art lossless techniques specifically devised for ECG signals. More precisely, the following metrics have been considered for comparison:

- *Compression Ratio (CR)* defined as the ratio between the number of bits before and after compression;
- *Complexity*: in particular we classified the algorithms into Simple, Feasible and Unfeasible. Simple algorithms are those based on linear predictors and encoding techniques which can be easily integrated in microcontrollers; Feasible algorithms can be also implemented in microcontrollers but they need more storage and computational resources; finally, Unfeasible algorithms are those with too high complexity to be integrated with low-cost microcontrollers, due to either complex predictors or huge memory requirements.

Complexity and the average compression ratio of the algorithms reviewed in Sec. II and the proposed algorithm are reported in Tab.I. In the case of the proposed algorithm, the full set of samples has been divided into 13000 blocks of $B = 50$ words each and for each block the devised algorithm has been applied.

As it is possible to observe in Tab.I, in camparison to other Simple solutions the proposed algorithm improves the compression ratio by at least 9%. Moreover, the CR of the proposed algorithm is higher respect to all the other solutions with the exception of [22]. However, as observed in Sec. II, the compression scheme proposed in [22] relies on a 6-stage Huffman encoder and a fuzzy predictor tuned using an off-line training scheme, thus it is too complex to be implemented in low-cost microcontrollers. Finally, it should be considered that differently from [29], [28] and [22], in the RAKE algorithm there are not parameters that must be tuned.

It is worth noting that a block size of $B = 50$ is compatible with the maximum payload length specified by several wireless communication protocols commonly used for IoT and WSN devices (i.e. IEEE802.15.4, ZigBee and BLE, to name just a few). Therefore a block can be processed by a microcontroller inside a wearable wireless monitoring system and sent to a second remote device that acts as a gateway and which can offer more storage resources. In this case only $B = 50$ words at a time have to be stored in the microcontroller and this number of words can be easily handled.

The proposed algorithm introduces a latency due to the fact that it operates on blocks of data. However, accordingly to the IEEE11073 standard [34], the maximum latency is lower than 500ms. In fact, considering a sampling frequency of 360Hz, about 50/360≈140ms are needed to acquire a full block and a few milliseconds are sufficient to process and send a block, considering a 4MHz microcontroller and a transmission rate of 250kbps. Therefore the overall expected latency is in the order of 150ms.

Finally, the reader could observe that when a block is lost than all successive samples cannot be reconstructed. However, this observation is true also for the other compression schemes discussed in this paper when a residue is lost or corrupted. A possible solution is to use forward error correction codes which improve reliability but reduce compression efficiency. For the sake of space we cannot further discuss this aspect.

ISBN 978-90-827970-1-5 © EURASIP 2018

Solution	Chen [26]	Lian [27]	Luo [28]	Li [29]	Deepu [30]	Chen [22]	Garg [31]	Proposed
Actual CR	2.43	2.25	2.53	2.45	2.38	2.84	2.2	2.67
Predictor	Adaptive	First order	Fuzzy	Adaptive	Adaptive	Fuzzy/PSO	Adaptive	Zero order
Encoder	Huffman (2-st.)	Dynamic Pack.	Huffman (2-st.)	VLC (1-st.)	Huffman (1-st.) + Dynamic Pack.	Huffman (6-st.)	Huffman (1-st.)	RAKE
Complexity	Simple	Simple	Feasible	Simple	Simple	Unfeasible	Simple	Simple

TABLE I
COMPARISON OF LOSSLESS COMPRESSION ALGORITHMS FOR ECG SIGNALS

V. CONCLUSION AND FUTURE WORKS

In this paper we have presented a simple and effective lossless compression algorithm for ECG signals. Using only a simple zero order predictor and counting operations, the algorithm is able to outperform several existing solutions. Moreover, considering its inherent low complexity and memory requirement, it is well suited for low cost ECG monitoring systems based on resource constrained IoT devices.

As future works we will investigate performance of the proposed algorithm with other types of biomedical signals.

REFERENCES

[1] W. H. Organization. Cardiovascular diseases (cdvs). [Online]. Available: http://www.who.int/mediacentre/factsheet/en/

[2] Y. Wang, S. Doleschel, R. Wunderlich, and S. Heinen, "A wearable wireless ecg monitoring system with dynamic transmission power control for long-term homecare," *Journal of medical systems*, vol. 39, no. 3, p. 35, 2015.

[3] S. Rafiq, A. H. M. Z. Alam, and M. R. Islam, "Development of ecg home monitoring system," in *2017 IEEE Regional Symposium on Micro and Nanoelectronics (RSM)*, Aug 2017, pp. 80–83.

[4] H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst, "Compressed sensing for real-time energy-efficient ecg compression on wireless body sensor nodes," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 9, pp. 2456–2466, 2011.

[5] S. Padhy, L. Sharma, and S. Dandapat, "Multilead ecg data compression using svd in multiresolution domain," *Biomedical signal processing and control*, vol. 23, pp. 10–18, 2016.

[6] G. Campobello, S. Serrano, A. Leonardi, S. Palazzo, "Trade-offs between energy saving and reliability in low duty cycle wireless sensor networks using a packet splitting forwarding technique," *EURASIP J. on Wireless Communications and Networking*, vol. 2010, pp. 1687–1499, 2010.

[7] B. Ramachandran and S. Bashyam, "Development of real-time ecg signal monitoring system for telemedicine application," in *2017 Third International Conference on Biosignals, Images and Instrumentation (ICBSII)*, March 2017, pp. 1–4.

[8] N. Dey, A. S. Ashour, F. Shi, S. J. Fong, and R. S. Sherratt, "Developing residential wireless sensor networks for ecg healthcare monitoring," *IEEE Transactions on Consumer Electronics*, vol. 63, no. 4, pp. 442–449, November 2017.

[9] G. Campobello, A. Segreto, and S. Serrano, "Data gathering techniques for wireless sensor networks," *Int. J. Distributed Sensor Networks*, vol. 2016, pp. 1–17, Mar. 2016.

[10] D. Salomon and G. Motta, *Handbook of Data Compression (5.ed.)*. Springer, 2010.

[11] K. Sayood, *Introduction to data compression (4.ed.)*. The Morgan Kaufmann series in multimedia information and systems, 2012.

[12] C. P. Antonopoulos and N. S. Voros, "Resource efficient data compression algorithms for demanding, wsn based biomedical applications," *Journal of biomedical informatics*, vol. 59, pp. 1–14, 2016.

[13] FDA. Guidance for industry: Diagnostic ecg guidance. [Online]. Available: http://www.fda.gov/RegulatoryInformation/Guidances/ucm073942.htm

[14] Association for the Advancement of Medical Instrumentation, AAMI, "Diagnostic electrocardiographic devices," *ANSI/AAMI EC111991/(R)*, 2001.

[15] C. J. Deepu, C. H. Heng, and Y. Lian, "A hybrid data compression scheme for power reduction in wireless sensors for iot," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 2, pp. 245–254, April 2017.

[16] G. Campobello, O. Giordano, A. Segreto, and S. Serrano, "Comparison of local lossless compression algorithms for wireless sensor networks," *J. Network and Computer Appl.*, vol. 47, no. C, pp. 23–31, Jan. 2015.

[17] K. C. Barr and K. Asanovic', "Energy-aware lossless data compression," *ACM Trans. Comput. Syst.*, vol. 24, no. 3, pp. 250–291, 2006.

[18] G. B. Moody and R. G. Mark, "The impact of the mit-bih arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, May 2001.

[19] M. Galeano, A. Calisto, A. Bramanti, S. Serrano, G. Campobello, and B. Azzerboni, "R-point detection for noise affected ecg recording through signal segmentation," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*. IEEE, 2010, pp. 638–641.

[20] S. M. Jalaleddine, C. G. Hutchens, R. D. Strattan, and W. A. Coberly, "Ecg data compression techniques-a unified approach," *IEEE transactions on Biomedical Engineering*, vol. 37, no. 4, pp. 329–343, 1990.

[21] S. Kiranyaz, T. Ince, and M. Gabbouj, "Real-time patient-specific ecg classification by 1-d convolutional neural networks," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 3, pp. 664–675, March 2016.

[22] S. L. Chen, M. C. Tuan, T. K. Chi, and T. L. Lin, "Vlsi architecture of lossless ecg compression design based on fuzzy decision and optimisation method for wearable devices," *Electronics Letters*, vol. 51, no. 18, pp. 1409–1411, 2015.

[23] C.E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

[24] D.A. Huffman, "A method for construction of minimum redundancy codes," *Proc. IRE*, vol. 40, pp. 1098–1101, 1952.

[25] S.W. Golomb, "Run-length enconding," *IEEE Trans. Information Theory*, vol. 12, no. 4, pp. 399–401, 1966.

[26] S.-L. Chen and J.-G. Wang, "Vlsi implementation of low-power cost-efficient lossless ecg encoder design for wireless healthcare monitoring application," *Electronics Letters*, vol. 49, no. 2, pp. 91–93, 2013.

[27] C. J. Deepu, X. Zhang, W.-S. Liew, D. Wong, and Y. Lian, "An ecg-soc with 535nw/channel lossless data compression for wearable sensors," in *Solid-State Circuits Conference (A-SSCC), 2013 IEEE Asian*. IEEE, 2013, pp. 145–148.

[28] G.-A. Luo, S.-L. Chen, and T.-L. Lin, "Vlsi implementation of a lossless ecg encoder design with fuzzy decision and two-stage huffman coding for wireless body sensor network," in *Information, Communications and Signal Processing (ICICS) 9th Int. Conference on*. IEEE, 2013, pp. 1–4.

[29] K. Li, Y. Pan, F. Chen, K. T. Cheng, and R. Huan, "Real-time lossless ecg compression for low-power wearable medical devices based on adaptive region prediction," *Electronics Letters*, vol. 50, no. 25, pp. 1904–1906, 2014.

[30] C. J. Deepu and Y. Lian, "A low complexity lossless compression scheme for wearable ecg sensors," in *2015 IEEE International Conference on Digital Signal Processing (DSP)*, July 2015, pp. 449–453.

[31] B. Garg, S. Yadav, and G. K. Sharma, "An area and performance aware ecg encoder design for wireless healthcare services," in *20th Int. Symposium on VLSI Design and Test (VDAT)*, May 2016, pp. 1–6.

[32] G. Campobello, A. Segreto, S. Zanafi, and S. Serrano, "Rake: A simple and efficient lossless compression algorithm for the internet of things," in *2017 25th European Signal Processing Conference (EUSIPCO)*, Aug 2017, pp. 2581–2585.

[33] Google. Protocol buffers: Encoding: Signed integers. [Online]. Available: https://developers.google.com/protocol-buffers/docs/encoding

[34] *IEEE 11073-00101 Health Informatics - PoC Medical Device Communication*, ISO/IEEE Std. 11 073, 2008.

ISBN 978-90-827970-1-5 © EURASIP 2018