Dynamic Alternation of Huffman Codebooks for Sensor Data Compression

Daniel Yunge, Sangyoung Park, Philipp Kindt, Samarjit Chakraborty

***Abstract*—Signal compression is crucial for reducing the amount of communication, and hence power consumption of wireless sensors. Lossless compression techniques, such as Huffman coding, are often used in healthcare applications since they do not compromise the integrity of vital signals. Techniques that adapt to changing signal patterns have been proposed. However, most of them involve significant computation overhead or are too simple to maintain high compression rates under changing signal patterns. In this paper, we propose a technique that makes use of multiple codebooks, which are generated offline based on the signal context. We observe that the symbols that compose a big variety of signals follow Laplacian distributions in which the spread changes over time. This can be effectively utilized to generate a set of codebooks. Then, appropriate codebooks are selected online depending on the currently measured spread, which ensures high compression efficiency and the adaptability to changing signal patterns. Our experiments on real-world datasets show that our approach is computationally very efficient, and exhibits competitive compression rates. Our proposed technique outperforms a state-of-the-art compression algorithm, FAS-LEC, in terms of average data reduction by 3.7%, while consuming a similar amount of energy. Compared to the adaptive Huffman method, which achieves near-optimal compression rates, our results indicate energy savings of 12% due to the reduced computational complexity, while the compression rate is reduced by less than 2.1%.**

I. Introduction

The rapid development of the Internet of Things (IoT) opens a new era in healthcare services, in which the use of inexpensive wearable medical sensors allows patients to be monitored remotely in their home environments, thereby reducing operational costs and hospital stays. The vast amount of data generated and transmitted from the wearable devices has motivated the use of compression algorithms in sensing devices, which reduce the communication costs and the energy consumption of the sensors and intermediate devices which transport the data into the cloud.

Due to the nature of vital healthcare applications that require high accuracies, lossless compression techniques such as Huffman coding are often used in these domains. Huffman coding reduces the size of a message (or signal) by redefining the bit length of the *symbols* that compose the message according to their frequencies of occurrence in the message. However, this method requires the frequencies of occurrence of the symbols before the message is processed. For that reason, a first pass over the message is required for gathering occurrence information of the symbols into a *codebook*, whereas in a second pass, the message is compressed. Unfortunately, the high computational cost of creating a codebook online, the need of sending the codebook to the decoding counterpart and the limitations on the number of samples which can be buffered, make the two-pass approach ineffective on live signals. On the other hand, an adaptive variant exists which constructs a Huffman codebook *while* it processes the sensor signals, without the need for performing encoding in two passes. As a consequence, neither signal buffering is required nor the transmission of a whole codebook from the encoder to the decoder, as the latter can construct the codebook inferred from the received symbols. However, the computational cost associated with the modification of the adaptive codebook grows quickly with each new compressed symbol, as this has the form of a binary tree, also known as Huffman tree.

In order to address this issue, a modified adaptive Huffman algorithm has been proposed, in which the leaves of the Huffman tree represent sets of symbols instead of individual symbols, reducing the size of the tree effectively [13]. However, as symbols are continuously being regrouped according to their changing frequencies of occurrence, the associated computation overhead and net energy savings remain uncertain. Another technique "trims" the Huffman tree by limiting the number of symbols allocated in the codebook [8]. Only a number of most frequent symbols are kept in the tree and coded using the adaptive method. Less frequent symbols, in contrast, are transmitted un-encoded. However, the evaluation in [8] showed the method to be ineffective on sensor signals having abrupt statistical changes, such as acceleration signals with phases of sudden motion and steadiness.

Another line of research has been conducted recently, which focuses on pre-allocating Huffman codebooks based on initial knowledge of the signal context. Pre-allocating a set of codebooks has benefits i) over static Huffman coding, since it allows quick adaptations on statistical changes in the sensor signals by switching among codebooks, ii) over adaptive Huffman coding, because it relieves the burden of maintaining a large Huffman tree, and thus less computation is required during run-time. An example is [9], where multiple pre-allocated code mappings are generated from the offline analysis of similar sensor data. However, this approach relies on analyzing the signals to be compressed offline and is therefore limited to them.

Another case of compression using context information is LEC, which makes use of a single table for creating codewords efficiently [6]. Codewords are built by appending a prefix of variable bit-length to the symbol to be encoded. The prefixes are looked up in the table and are associated with a range of values in which the symbol is found. Later on, more adaptability was aggregated to FAS-LEC [15], which proved to be more effective than LEC. FAS-LEC rotates the prefix table in order to make the shortest prefix match the most frequent symbol range observed in the stream in the past. In addition, it splits the table into two parts, which can rotate independently. Although FAS-LEC proved to be computationally light-weight and provides compression efficiencies up to 95% on meteorological datasets [15], its effectiveness on signals from other

The authors are with the Chair of Real-Time Computer Systems at Technical University of Munich, Arcisstrasse 21, 80290, Munich, Germany. E-mail: {yunge,park,kindt,chakraborty}@rcs.ei.tum.de

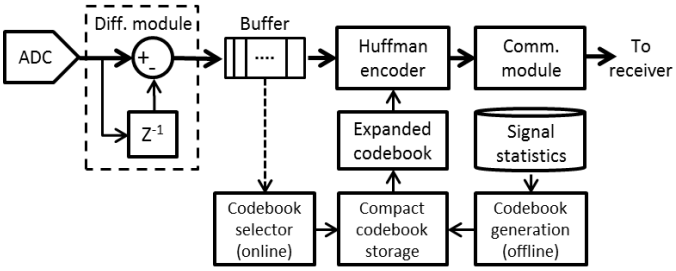

Fig. 1: Encoding part of the proposed scheme.

source has not been verified. Accordingly, we included this method in the evaluation section.

In this paper, we propose a Huffman coding technique for compressing sensor signals using multiple pre-allocated codebooks, which are alternated according to the changing variance of the symbols observed. Our work takes a cue from [9], however, we introduce major changes, which significantly enhance the practicality of the idea. First, we identify the reason for the previous approach being less effective, which is the lack of a differentiation process of adjacent samples. As a consequence, this method is prone to offsets in the sensor signals. If a codebook generated for a particular offset is used for compressing a signal having an at least slightly different offset, the compression performance is harmed. Accordingly, we compress the difference between consecutive samples in our proposed method, similar as in [5]. Then, the resulting stream of differentiated samples is coded using pre-allocated codebooks generated from Laplacian distributions with different spreads. In contrast, the procedure to generated codebooks in [9] has no parametrization and relies on previously recorded signals. Finally, in our proposed method, codebooks are alternated during run-time according to an alternation policy designed to increase compression performance and minimize energy consumption. As mentioned, an optimal codebook is selected directly from the observed variance of the symbols, in contrast to [9], in which all pre-allocated codebooks are evaluated online for the selection.

In the following section, we provide an overview of our proposed technique, elaborate how the codebooks are generated, stored, and alternated at low computational cost. Then, we compare our method with state-of-the-art compression algorithms for wireless sensors using real-world medical and meteorological datasets. Our results show that our method achieves the highest compression rates for medical datasets (acceleration, EEG and ECG), achieving on average 4.3% and 0.6% communication reductions compared to FAS-LEC and the Adaptive Huffman algorithm, respectively. These increased compression rates are achieved with a similar energy consumption to FAS-LEC and by a 19% reduced energy consumption compared to adaptive Huffman compression.

II. Multiple Static Huffman Codebooks

A. Overview

The overall scheme is summarized in Fig. 1. The difference of adjacent raw signal samples from the ADC is calculated using the differentiation module and stored into a buffer. Then, the buffered signal is analyzed by the codebook selector block to decide on whether to alternate the codebook or not. If an alternation takes place, one of the multiple codebooks allocated

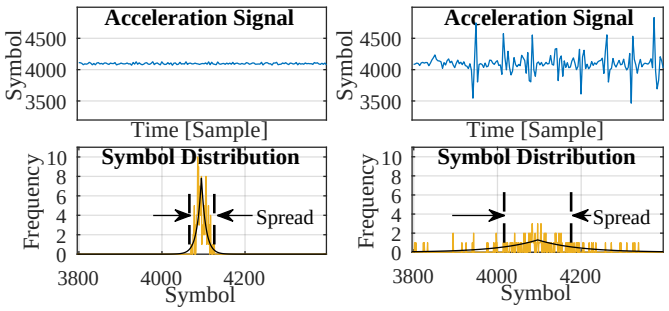

Fig. 2: Symbol distribution of two acceleration signals after differentiation process compared to a Laplacian function.

in the codebook storage block is selected and expanded, since they are stored using a compact format for memory saving purposes. The expansion process incurs computation and communication costs, therefore the codebook selector block evaluates whether the estimated cost and data reductions justify a codebook alternation. Once the codebook is expanded, buffered samples are encoded quickly into codewords by a simple table lookup. Finally, the variable bit-length codewords are concatenated and passed on to the communication module for wireless transmission.

B. Storage of Codebooks

In what follows, we describe more details on the codebook format and coding process. Basically, the expanded codebook is a double-column lookup table, whose rows represent the symbol to be encoded. An encoded symbol is known as a *codeword*, and is composed of two values which are allocated in each column of the lookup table: content and bit length. Accordingly, a codeword is created from the value found in the content cell of the corresponding row in the codebook, and padded with zero-bits to achieve the bit length specified in its length cell. For example, let the content cell and length cell of row 20 be the values 255 and 15, respectively. As a result, symbol 20 is represented by the bit sequence "000 0000 1111 1111", which was generated from the bit sequence "1111 1111" (255) and left-padded with 7 zeros to achieve the bit length of 15.

This codebook scheme is not part of the Huffman algorithm, which uses a binary tree structure, but is an adaptation described in [11] based on the canonical Huffman algorithm [12], a variant of the classical method. It also changes the way in which codewords are constructed, as described in what follows. The canonical method first generates an initial set of codewords using the classical Huffman algorithm, from which only the codewords' bit length information is used and the content information is removed. Then, it uses an algorithm that establishes new content for the codewords, which are always a bounded value between 0 and the maximal value found in the original symbol set, e.g. 8191 for a 13 bits signal. As a result, codewords can be stored in a table using the double-column scheme explained above, which is considerably more efficient than the binary tree structure [11].

We have extended this scheme by pre-allocating multiple codebooks in memory and making the compression system alternate among them during run-time. This imposes the need for a compact format of the pre-allocated codebooks. More precisely, because consecutive values in the content and length

columns have a high probability of being repetitive, codebooks were reduced (offline during design-time) by applying run-length encoding to these values. In the case of the codebook set used for the evaluation in Section III, for example, an expanded codebook consisting of 8192 rows and two columns of two and one bytes, respectively, can be reduced from 25 kB to only 222 bytes in its compact format.

C. Generation of Codebooks

Finding a set of codebooks which compresses a data stream optimally is an NP-hard problem [7], which is hard to solve even offline. However, we will show that for a large set of different real-world sensor signals, it is feasible to find a reduced codebook set which has compression rates near to the optimum ($\pm 1\%$). This is possible because most sensor-signals exhibit significant information redundancy between temporal adjacent samples, which can be reduced by a differentiation process, i.e., calculating the difference between consecutive samples prior to compressing them. The result is that, after the differentiation process, the highest frequency of occurrence is that of symbol zero, and the resulting symbol distribution corresponds well with some distributions of the exponential family, like Gaussian and Laplacian. As with other continuous signals (e.g., as for speech [4] and acceleration [5]), we found that the Laplacian distribution achieves better compression rates on the evaluated datasets. The concept of symbol distributions is depicted in Fig. 2, assuming acceleration signals as an example.

During design-time, using the Laplacian function with different spreads – represented by the diversity parameter $B = \frac{1}{\sqrt{2}\sigma}$ – a set of codebooks is generated and allocated in the memory of the compressing device. During run-time, the codebook selector block in Fig. 1 matches the buffered samples to a given allocated codebook by means of a maximum likelihood estimation of the diversity parameter, $\hat{B}$, which is the mean absolute deviation (MAD) of the buffered samples [4]. If the codebook associated to the particular estimator does not exist, the codebook with closest diversity parameter is selected as the candidate.

D. Codebook Alternation

In principle, a codebook is alternated whenever the observed diversity parameter $\hat{B}$ changes. However, skipping a codebook alternation is worthwhile if $\hat{B}$ is found to be too close to that of the previous set of buffered samples, as the codebook expansion process incurs significant computation efforts. Skipping alternations, along with increasing the buffer size, are important measures to reduce the average execution time of our proposed method without compromising significantly its compression performance. Moreover, in addition to the processing overhead, a codebook alternation also incurs some communication overhead for notifying the decoding counterpart about the new codebook to be used. Accordingly, minimal codebook proximity for an alternation and the buffer size values must be found to minimize energy consumption.

III. Experimental Results

In this section, we compare our compression method with others by evaluating their compression performance and associated energy consumptions of the radio and MCU. The evaluated methods are the adaptive Huffman coding, FASLEC and ours. In addition, the uncompressed transmission was also evaluated. The evaluation was carried out using different datasets containing continuous signals. Acceleration signals associated with physical activity were used from [10], a dataset for activity recognition research, generated from multiple body-worn sensors. In addition, weather data (i.e., temperature, humidity, wind speed and solar radiation) from [14] was considered. Finally, EEG signals obtained from [1] and ECG signals from [2], [3] were evaluated. Detailed information on the signal characteristics, such as resolution, sampling rate, calculated entropy and optimal compression rate can be found in Table I. In addition, the effective number of codebooks used by our method and the number of codebook alternations are shown in the table under CB_n and Alt_{cb}, respectively. The length of each dataset was 500.000 samples.

TABLE I: Characteristics of the evaluated datasets.

	Res bits	S_r Sa/s	Epy bits/sym	R_{opt}	MultiStatic CB_n	MultiStatic Alt_{cb}
Accel	13	33	-8.2	0.37	16	2050
Temp	11	1/300	-3.4	0.69	10	283
Humid	8	1/300	-2.0	0.76	5	134
Wind	8	1/300	-2.2	0.73	7	202
Solar	11	1/300	-4.4	0.60	21	3609
EEG	13	512	-5.0	0.61	13	168
ECG	13	1000	-6.3	0.52	14	890

An STM32F401 Nucleo board running an ARM Cortex-M4 MCU at its default frequency of 16 MHz was used to evaluate the algorithms. To evaluate the compression performance and energy consumption of the methods, uncompressed samples were transferred from a PC to the MCU at a rate of 1 kSa/s using a USB-to-Serial link. After the samples were compressed by the MCU, the compressed stream was passed to a Bluetooth Low Energy (BLE) communication module in chunks of 20 bytes. A wireless link was established from the BLE module of the sensor to a BLE USB-dongle connected to a PC. At the end, the compressed stream was stored in a file for evaluation purposes, and decompressed and compared with the original file for a later evaluation. The current consumptions of the MCU and communication module were acquired using an NI-PXIe-6124 DAQ at 100 kSa/s.

Our proposed method used a set of 21 compact static codebooks for all datasets, which were generated from a Laplacian function with diversity parameters between 0.5 and 180. For the sake of fairness, the lengths of the codebooks were related to the analyzed signals, i.e., 256-symbol codebooks were used for 8-bit signals, 2048 for 11-bit and so on. A buffer size of 100 samples was defined, and a distance of 2 codebooks in the set between the current and the candidate codebook was required to trigger an alternation event. In terms of memory requirements, the complete MCU firmware used 18 kB of ROM, 4.7 kB of which were used by the compact codebook set. On the other hand, 3 kB of RAM were used by the firmware, in addition to that dedicated to the expanded codebook (25 kB, 6 kB, or 0.5 kB when compressing 13-bit, 11-bit or 8-bit signals, respectively).

The results of the evaluation in terms of compression efficiency are shown in Fig. 3. The compression efficiency is defined as $\eta = size_{opt}/size_{eval}$, where $size_{opt}$ is the data volume after optimal compression according to Shannon's entropy formula [6], and $size_{eval}$ is the data volume generated by the evaluated compression method. It can be observed that

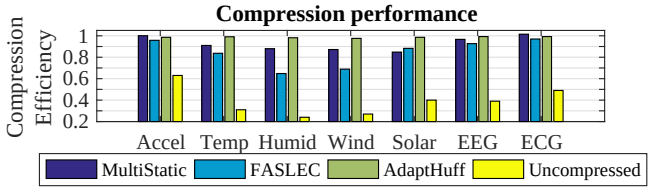

Fig. 3: Compression efficiency of the evaluated algorithms on 7 different signal types, using a buffer size of 100 samples.

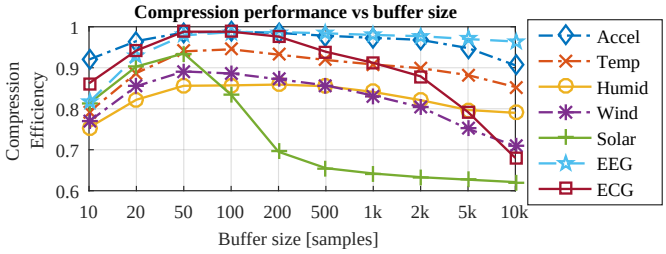

Fig. 4: Compression efficiency of the evaluated algorithms using different buffer sizes.

our method – MultiStatic – performs very close to the optimum (above 97% of compression efficiency) on signals which have high entropy, and above 85% for the other signals. The average compression efficiencies on all signals are 93%, 84% and 99% for our proposed method, FAS-LEC and the adaptive Huffman method, respectively. The compression deficiency of FAS-LEC on low entropy datasets (≈2 bits/sym) is caused by the fact that the prefix table proposed by [15] generates codewords which have a minimum length of 3 bits.

In addition, an analysis of the effects of different buffer sizes on the compression efficiency was conducted. The results are presented in Fig. 4. It can be observed that the compression efficiency achieves its maximum around buffer sizes between 50 and 100 samples. This can be explained as follows. For short buffers, short-term variations of the signal lead to a non-zero mean of the difference symbols and in addition, more codebook alternations occur. On the other hand, large buffer sizes result in more signal patterns being compressed with a single, non-optimal codebook.

The energy consumptions of the MCU, communication module and their sum are shown in Fig. 5 for every dataset and method considered. Along with the energy due to the compression process, the measured energy includes the associated

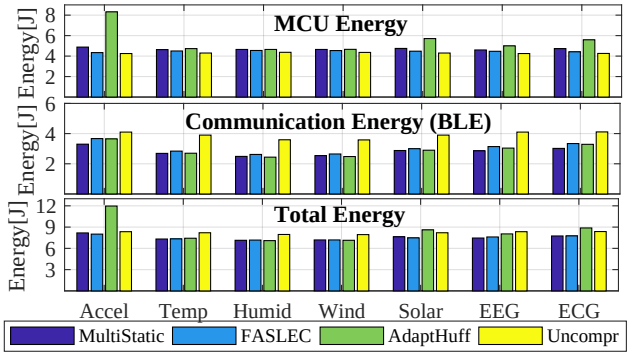

Fig. 5: Energy consumption evaluation of the communication module, MCU and total.

overhead processes of the MCU and communication module, as well as the energy consumed during sleep periods. Considering only the MCU energy, our proposed method consumes 12% less energy than adaptive Huffman compression, but 5.2% and 9.3% more energy than FAS-LEC and uncompressed transmission, respectively. Considering the communication energy, on the other hand, the proposed method achieves energy savings of 6.7% with respect to FAS-LEC, 2.8% with respect to adaptive Huffman and 28% with respect to uncompressed transmission. Finally, it can be observed that the proposed method and FAS-LEC achieved the same total energy consumption (7.5[J]) for the given setup, below that of adaptive Huffman (8.4[J]) and uncompressed transmission (8.2[J]).

IV. Conclusions

In this paper, we proposed a Huffman codebook alternation technique to compress sensor signals, which makes use of pre-allocated codebooks generated at design time. We have presented a practical implementation of the proposed technique and performed a comparison with other compression schemes on medical and meteorological datasets, based on real-world measurements. The proposed technique is adaptable to signal pattern changes and computationally efficient, such that it achieves a similar compression rate to adaptive Huffman coding with considerably less amount of computation, comparable to FAS-LEC.

References

[1] R. G. Andrzejak et al. Nonrandomness, nonlinear dependence, and nonstationarity of electroencephalographic recordings from epilepsy patients. *Phys. Rev. E*, 2012.

[2] R. Bousseljot et al. Nutzung der ekg-signaldatenbank cardiodat der ptb über das internet. *Biomed. Technik/Biomedical Engineering*, 1995.

[3] A. L. Goldberger et al. Physiobank, physiotoolkit, and physionet. *Circulation*, 2000.

[4] S. Kotz, T. Kozubowski, and K. Podgorski. *The Laplace distribution and generalizations: a revisit with applications to communications, economics, engineering, and finance*. Springer Science & Business Media, 2012.

[5] Y. Liang and W. Peng. Minimizing energy consumptions in wireless sensor networks via two-modal transmission. *SIGCOMM Comput. Commun. Rev.*, 2010.

[6] F. Marcelloni and M. Vecchio. An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks. *The Computer Journal*, 2009.

[7] M. Mitzenmacher. On the hardness of finding optimal multiple preset dictionaries. *IEEE Trans. on Inf. Theory*, 2004.

[8] A. Reinhardt et al. Trimming the tree: Tailoring adaptive huffman coding to wireless sensor networks. *European Conf. on Wireless Sensor Networks (EWSN)*, 2010.

[9] A. Reinhardt et al. Pre-allocating code mappings for energy-efficient data encoding in wireless sensor networks. In *IEEE Int. Conf. on Perv. Comp. and Comm. Workshops (PERCOM Workshops)*, 2013.

[10] D. Roggen et al. Collecting complex activity datasets in highly rich networked sensor environments. In *Int. Conf. on Networked Sensing Systems (INSS)*, 2010.

[11] M. Schindler. Practical huffman coding, 1998.

[12] E. S. Schwartz and B. Kallick. Generating a canonical prefix encoding. *Commun. ACM*, 7(3), 1964.

[13] C. Tharini and P. V. Ranjan. Design of modified adaptive huffman data compression algorithm for wireless sensor network. *Journal of Comp. Science*, 2009.

[14] UMassTraceRepository. http://traces.cs.umass.edu/.

[15] M. Vecchio et al. Adaptive lossless entropy compressors for tiny iot devices. *IEEE Trans. on Wireless Commun.*, 2014.