An Approximate Compressor for Wearable Biomedical Healthcare Monitoring Systems

Farzad Samie, Lars Bauer, Jörg Henkel
Chair for Embedded Systems (CES)
Karlsruhe Institute of Technology (KIT), Germany
{farzad.samie, lars.bauer, joerg.henkel}@kit.edu

ABSTRACT
Technology advancements as well as the Internet-of-Things paradigm enable the design of wearable personal healthcare monitoring systems. Ultra-low-power design is a challenging area for these battery-operated wearable devices, where the energy supply is limited and hardware resources are scarce. Some biomedical applications tolerate small errors in the values of the biosignal or small degradation in the quality, which can be exploited to reduce the energy requirements.

This paper presents an approximate compression technique for biosignals in a wearable healthcare monitoring system. It takes advantage of error tolerance in biosignals and finds the shortest code to compress the data while keeping the error in an acceptable range. Our approximate compressor does not demand any hardware modification and thus can be used in existing wearable devices. The proposed approach for reducing the size of the Huffman table can save 1 MBit storage, on average. It also makes our approximate compressor suitable for runtime adaptation, i.e. creating a new Huffman table based on updated values.

Compared to state-of-the-art, our experimental results show up to 60% reduction in data size that is to be transmitted via radio. As wireless communication contributes significantly to the total energy consumption of wearable devices, this improvement can increase the battery lifetime of our healthcare monitoring prototype from 7 days to 10 days.

Keywords
Biomedical Healthcare, Wearable Devices, Approximate Computing, Compression, Internet of Things, IoT

1 Introduction
Recent advances in technology of embedded processors, sensors, and wireless communication have enabled the design of compact, ultra-low power, low cost embedded devices that can be interconnected as the key components of the Internet of Things (IoT) [1]. IoT is covering an ever-increasing range of applications including assisted living, smart buildings and, of particular interest, applications for healthcare. Wearable healthcare monitoring systems are used to monitor elderly people or patients’ health status while they are out of the hospital doing their daily activities. Wearable devices are also very popular and useful for sports, fitness and wellness to measure daily activity, sleep patterns, and other parameters related to well-being [2]. They sense and process the vital signals of a person (e.g. ECG, EEG, EMG) anywhere and anytime as long as it is needed, and transmit the data to the cloud server via smartphone or directly

Figure 1: Simplified architecture of an IoT-based wearable ECG monitoring device

via a WiFi link. Since these devices are battery-operated, they need to be ultra-low power [3].

Let us consider an IoT-based wearable healthcare monitoring device that needs to be carried out for a long period of time in order to record and transmit patient’s biomedical data such as electrocardiogram (ECG) for subsequent clinical diagnosis [4, 5]. Figure 1 shows such an architecture for wearable ECG monitoring. After the retrieved signal is filtered and digitized, the ECG signal is compressed to be transmitted to a smartphone or a cloud server for detailed analysis of the morphology of ECG. Characteristic features are extracted and then classification is performed to detect abnormalities. The smartphone can also perform real-time delineation and classification of heartbeats for some applications such as heart rate variability and arrhythmia classification [6]. As the radio component accounts for the largest energy consumption share on these devices [7], reducing the amount of data to be transmitted is essential to extend the battery lifetime [6, 8].

Figure 2a shows three seconds of an ECG signal (left) and the histogram of ECG values recorded for 2 hours, 10 minutes and 12 seconds (right). The ECG signal values have a Gaussian distribution, which indicates that some values (near zero) occur noticeably more often than the others. Gaussian distributions are popular candidates for being compressed by Huffman coding, which assigns short codes to high frequently occurring values and long codes to those that occur infrequently [4, 6, 9]. Moreover, in a wide range of applications and systems, instead of transferring an entity (file, frame, value, etc.) in its entirety, only the difference with a reference (e.g. its previous value, a fix entity, or its previously transmitted entity) is sent. Delta coding is also advantageous for ECG signals and can be exploited for storing and transmitting them [10, 11]. Instead of sending absolute
and 80 and thus be encoded by only 11 and 12 bits, which

This example demonstrates a significant potential for
compression, and consequently, energy saving. Accordingly, this
motivates our key idea of approximate compression: a
novel data compression scheme for inherently error resilient
signals such as biomedical signals (e.g. ECG) that can tol-
erate errors as long as the error amplitude is small enough
and subsequent data processing (e.g. ECG delineation) is
still possible. As we will show, approximating successive
delta values may result in an aggregated error in the re-
constructed signal, which can lead to an unacceptable error
range. Therefore, we propose a method to keep the error in
an acceptable range.

The novel contributions of this paper are as follows:

• We introduce an approximate computing approach for
achieving higher compression ratio for biomedical sig-

Paper structure: in Section 2, we briefly review related work
on biosignal compression and approximate computing. Then
we provide a detailed presentation of our novel algorithm in
Section 3. Experimental results are presented and discussed
in Section 4, while Section 5 concludes the paper.

2 Related Work

2.1 Biomedical Signal Compression

A compression scheme for mobile healthcare monitoring de-
vices is presented in [6]. Considering the periodicity of the
biomedical signals, the technique is inspired by video com-
pression standards (e.g. MPEG) where video is divided into
primary frames as references, and secondary frames which
are constructed by subtracting the current frame from the
primary frame. Although biomedical signals such as ECG
are periodic by nature, their period may vary with time as
the activity of the person changes. An alignment technique
(e.g. sequence alignment) is needed to match two frames,
which introduces high computational cost.

Several transform-based data compression techniques have
been proposed for biomedical systems. These techniques an-
alyze the signals in the transformed domain (e.g. frequency
domain), preserve the coefficients of the particular features
in the data, and truncate insignificant components without
loss in major information. The transform-based methods
include Discrete Wavelet Transform (DWT) [12], Discrete
Cosine Transform (DCT) [4, 8], and Discrete Fourier Trans-
form (DFT) [13]. However, these techniques are not suit-
able for wearable or embedded medical devices due to their
high computation demand. Most wearable medical devices
have limited computational capabilities and low throughput.
And even if they would have sufficient computational per-
formance, using these techniques would increase the energy
consumption significantly [6].

Ref. [10] has proposed a signal compression method for ECG
Holter systems. A quad level vector (QLV) is used to indi-
cate the ECG complexes and its information level. Then,
a 4-bit-wise Huffman coding method is exploited to dis-


Table 1: Code word length for some delta values

<table>
<thead>
<tr>
<th>Delta Value</th>
<th>77</th>
<th>78</th>
<th>79</th>
<th>80</th>
<th>81</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Length [bit]</td>
<td>11</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>

Similar to [10], the authors in [14] presented a compression

technique which includes multilevel vector (MLV) compres-
sion, an integer-linear programming (ILP) formulation, and
Huffman coding. The ILP problem represents a minimum
set-covering problem which chooses some signal samples to
be included in the compressed output while the others are
just discarded. The ILP problem is not only computationally
intensive and inapplicable for wearable devices, but also
makes the compression lossy which may change the form and
features of the signal after reconstruction.

Compression techniques that use prediction methods are
presented in [5, 15]. The prediction difference is then en-
coded using a Huffman table with a variable length coding.
The ECG signal is classified into four regions and for each

Key Observation: Table 1 shows the code word length for
some delta values compressed by Huffman coding. For ex-
ample, the code word length for delta values 77 is 11 bits,
while the code length of 78 is 14 bits. Assuming that the
actual data can accept a slight variation of ±1 (which is the
case for ECG), delta values 78 and 81 can be replaced by 77
and 80 and thus be encoded by only 11 and 12 bits, which
can save 3 and 2 bits, respectively.
region a suitable predictor is used. In [15], two Huffman tables are used, each of which covers a range of values.

In [11], a compression method for ECG signals is presented which is based on local extreme extraction, adaptive hysteretic filtering and LZW coding. The authors exploit a nonuniform sampling approach that keeps the local extremes such that most parts of the signal can be reconstructed.

Some of these existing techniques are orthogonal to our proposed techniques and can be used along with ours. For instance, our approximate compression can be applied to [6] for compressing the residual frames. Moreover, other Huffman-based techniques such as [11, 12, 15] can also benefit from our approximate approach. On the contrary, QLV [10] is not able to benefit from approximation in Huffman compression, because it uses Huffman codes as the delimiters. We compare our proposed technique with [10], and we show its effect when applied on top of [6] in Section 4.

2.2 Approximate Computing

Approximate computing is an emerging design paradigm that relaxes the conventional requirement of exact equivalence between the specification and implementation of a computing system. For error-tolerant applications, this paradigm allows the system to trade the output quality for computational effort (e.g., energy, delay, etc.) [16, 17]. Approximate computing techniques have developed at different parts and various layers of the computing stack.

**System level:** In [18], a design flow to study the inexactness at system level is presented in which the properties of individual inexact components are taken into account to estimate the inexactness of the entire system. Authors applied their technique to a real-life ECG application for QRS detection.

**Processing units:** There are many attempts to design approximate hardware units for specific arithmetic components (e.g., adders [19, 20, 21]), multipliers [22], 2-D DCT [23], etc.). A digital signal processing unit is proposed in [24] which uses transistor-level simplification to build approximate adders that can be used in DSP blocks.

**Memory:** Bortolotti et al. [25] present a hybrid memory architecture for low-power multi-core biosignal processors. The proposed architecture can operate at aggressive voltage scaling while tolerating memory errors. It trades the signal quality for improvement in energy saving while preserving correctness for critical data structures. In [26], authors propose techniques that enable applications to store data approximately for both persistent data and transient data in main memory.

Those approximate computing approaches not only require changes in the hardware design, but they also cannot solely guarantee the error range at the output. Additionally, their improved processing- and storage-operations do not provide significant energy savings in wearable devices, because wearable devices do not perform many of these operations, but their energy consumption largely depends on the wireless communication volume.

**Software applications:** Some applications (e.g., linear/non-linear systems of equations and combinatorial optimization problems) may employ iterative methods to find the output. In these iterative methods, a sequence of improving approximate solutions are generated before reaching the final converged solution. ApproxIt [27] is a framework for such applications that guarantees the quality of solution at algorithm-level. Ref. [28] proposes a model for approximate programming that allows the programmer to explicitly delineate information flow from approximate program component to precise component. Data structures can be declared as approximate or precise where approximate data is processed more cheaply and less reliably.

None of the aforementioned approximation approaches provide opportunities to reduce the size of transmitted data (i.e., to reduce the power consumption of the wireless communication), as they aim at reducing the complexity of computational intensive applications.

3 Our Approximate Compression

3.1 Error Tolerance

Assuming that the analog-to-digital converter (ADC) has W bits resolution, the ECG values can be represented by integer values in the range \([-2^{W-1}, 2^{W-1}-1\)]. Let \(s_i \in \mathbb{Z}\) denote the \(i^{th}\) ECG sample, \(-2^{W-1} \leq s_i \leq 2^{W-1}-1\), and let us assume that for the target application an approximated value \(s_i' = s_i + \varepsilon_i\) is accurate enough as long as \(\varepsilon_L \leq \varepsilon_i \leq \varepsilon_H\), where \(\varepsilon_L, \varepsilon_H \in \mathbb{Z}\) are the lower and upper bound of tolerable error, respectively (e.g., \(\varepsilon_L = -2, \varepsilon_H = 2\) for a tolerable error of \(\pm 2\)). This tolerable error is defined and given by the application and system designer for which the requirements of system for detection accuracy, robustness, etc. is taken into account. Given the basic definitions of delta codes \(i\) and approximated values \(ii\) on the left side of Eq. (1), the right side shows that the delta value can tolerate the same error \(\varepsilon_i = s_i' - s_i\) (\(\varepsilon_L \leq \varepsilon_i \leq \varepsilon_H\)) that was specified for the ECG samples. Thus, we can use an approximate delta value \(d_i' = d_i + \varepsilon_i\) instead of the exact delta value \(d_i\).

\[i) \quad s_i = s_{i-1} + d_i\]
\[ii) \quad s_i' = s_i + \varepsilon_i = s_{i-1} + d_i + \varepsilon_i\]
\[s_i' - s_i = d_i' - d_i \quad (1)\]

This offers the opportunity to exploit the error tolerance of delta values by intentionally adding a permitted error to a delta value such that the resulting approximated delta value has a shorter Huffman code. This is shown in Eq. (2), where \(c(d)\) represents the Huffman code length of a delta value \(d\).

\[d_i' = d_i + \varepsilon_i : \quad c(d_i') = \min_{\varepsilon \in \varepsilon_L, \varepsilon \leq \varepsilon_H} (c(d_i + j)) \quad (2)\]

However, the calculation \(s_i' = s_{i-1} + d_i'\) in Eq. (1) assumes that the exact previous absolute value \(s_{i-1}\) is still available when decoding the approximated value \(s_i'\). In practice, to be able to actually benefit from the shorter Huffman codes, only the approximated value \(s_i'\) will be transmitted and thus available when decoding the delta value. The entire flow is shown in Figure 3. The approximate compressor first calculates the delta value \(d_i\), applies an intentional error \(\varepsilon_i\) to obtain the approximated delta value \(d_i'\) and then either directly transmits its Huffman code \(c_i\) or temporarily stores it in a non-volatile memory (e.g., in case the wireless connection could not be established). The receiver decodes the Huffman codes into the approximated delta values \(d_i'\) and
Figure 3: The scheme of our compression and coding method

Figure 4: Approximating delta value with an error in acceptable range, and updated error range

then calculates \( s'_i = s'_{i-1} + d'_i \) to retrieve the approximated ECG values \( s'_i \).

The problem with this procedure is that it will lead to accumulated errors. For instance, if a particular sequence of approximated delta values always uses the largest tolerated error (i.e. \( \forall i : \varepsilon_i = e_H \) then this will lead to an error runaway of the absolute reconstructed value. Let us consider a sequence \( (d'_1, d'_2, \ldots, d'_i) \) of approximated delta values \( d'_i = d_i + \varepsilon_i \). Eq. (3) calculates the error \( E(s'_i) \) when comparing the approximated sample \( s'_i = s'_{i-1} + d'_i \) with the exact sample \( s_i = s_{i-1} + d_i \). The obtained recursion shows how errors are accumulated.

\[
E(s'_i) = s'_i - s_i = (s'_{i-1} + d'_i) - (s_{i-1} + d_i) \\
= (d'_i - d_i) + (s'_{i-1} - s_{i-1}) \\
= \varepsilon_i + E(s'_{i-1}) \tag{3}
\]

In the following, we present an effective scheme to guarantee that the reconstructed ECG sample \( s'_i \) is always in the given tolerance range \([e_L, e_H]\) despite the error accumulation. Let us assume that \( E(s'_{i-1}) \) is within the tolerated error range (by using our scheme), i.e. \( E(s'_{i-1}) \in [e_L, e_H] \). Then the question is which error is tolerable for the next sample. Eq. (4) shows how the bounds for the error \( \varepsilon_i \) of the next sample need to be modified, depending on the particular error \( E(s'_{i-1}) \) that was accumulated so far.

\[
\varepsilon_L \leq \varepsilon_i + E(s'_{i-1}) \leq e_H \\
\Rightarrow (\varepsilon_L - E(s'_{i-1})) \leq \varepsilon_i \leq (e_H - E(s'_{i-1})) \tag{4}
\]

Figure 4 shows an example where the tolerable error range is \( e_L = -2 \) and \( e_H = +2 \). The acceptable error that leads to the shortest Huffman code for this delta value is \( \varepsilon = -1 \). Therefore, the error range should be updated to \( e_L = -2 \) and \( e_H = +2 \).

3.2 Reducing Computational Overhead

Figure 5 shows the flow of approximate compression. For each delta value, depending on the error range, the adjacent Huffman codes are taken into account to find the shortest code. We propose a technique to reduce the overhead introduced to find the shortest code.

As shown in Eq. (2), finding the the best approximate delta value involves \((e_H - e_L + 1)\) memory reads accessing the Huffman table, as well as a total of \((e_H - e_L)\) comparisons to find the minimum code length. This must be done for every ECG sample, which would not only decrease performance and throughput, but would also cost energy. Therefore, we propose a scheme to reduce this overhead, resulting in a more efficient approximate compressor architecture in which only one memory access is required for compressing one delta value.

While the original approximation scheme needs to search and find the shortest code in the tolerated error range for every delta value, we use a modified Huffman table that contains the pre-computed shortest codes for the tolerated error. Let us first assume that the error range for each individual sample would be constant, i.e. \( \varepsilon \in [e_L, e_H] \). The new table has two entries for each delta value \( d_i \):

- shortest code in the tolerated error range \( sc_i \):
  \[
  sc_i = \arg\min\{cl(d_i + j)\}_{e_L \leq j \leq e_H}
  \]
- bias \( b_i \): the difference between \( d_i \) and the delta value whose Huffman code has been chosen as \( sc_i \).

Basis values keep track of the amount of tolerable error used by a particular delta value. This is required to ensure that subsequent delta values remain within the tolerable error, as will be explained in the following subsections.

Figure 6 provides an example of how our modified Huffman table is created. It shows some delta values and their corresponding Huffman codes on the left side (i.e. the original Huffman table). Then for each delta value the shortest Huffman code in the error range \([-2, +2]\) is chosen. For delta values \(-1, 0\), and 1, the shortest code is the original Huffman code.

\[
\begin{align*}
\text{ECG} & \quad \text{Error range} \\
\quad & \quad \text{Update Error range} \\
\quad & \quad \text{Find Shortest Code} \\
\quad & \quad \text{Default Error range} \\
\quad & \quad \text{Store/Transmit}
\end{align*}
\]

\[
\begin{align*}
\text{Huffman Table} & \quad \text{T}[d+e_L] \\
\quad & \quad \text{T}[d+e_H]
\end{align*}
\]
code, which means that these delta values are used with no approximation and hence, their corresponding bias values are 0. On the contrary, for delta value 3 the Huffman code that corresponds to delta value 1 is used. Thus, the delta value for 3 is approximated with a bias of -2.

3.2.1 Online and offline table construction

If the original Huffman table is known in advance, the shortest code (SC) table can be precomputed offline and stored in the memory. Alternatively, it could be built at runtime by doing the computation once and only once for each delta value. When a delta value appears for the first time, then its shortest code SC and bias value b would be computed by taking its adjacent delta values in the permitted range into account. The computed SC and b values would then be stored in the table for the next references to this delta value.

3.2.2 How to use our Approximate Huffman Table

As explained in Section 3.1, the tolerable error range $[e_L, e_H]$ changes for successive approximate delta values. However, to construct the SC table in Figure 6 we assumed a constant changes for successive approximate delta values. However, if the first delta value is 2 then the bias would be

$$E_L - E_{L-1} = 2$$

As an example, consider the SC table shown in Figure 6. If the first delta value is 2 then the bias would be -2. If the next delta value is 5, we have to investigate the delta value range 5 – 1 to 5 + 3. When we look for the entry of 5 – (-1) = 6 in the SC table, as a matter of fact, we cover the delta value range of 6 – 2 to 6 + 2 (see Figure 7). Algorithm 1 describes how to use the SC table at runtime for our proposed approximate compression.

![Index addressing to keep the default SC table usable for successive approximation. Accumulated error is -2, and the error range is ±2.](image)

**Algorithm 1:** Compressing operation using shortest code table

1. $Sp \leftarrow$ getSample(); // get the first signal’s sample
2. bias $\leftarrow$ 0; // initial bias value is zero
3. while (True) do
4. $Sn \leftarrow$ getSample(); // new sample
5. $d \leftarrow$ $Sn$ - $Sp$; // delta value
6. $Sp \leftarrow$ $Sn$; // previous sample
7. $sc \leftarrow sc_{table}.get\_code(d-bias)$; // get shortest code
8. $bias \leftarrow sc_{table}.get\_bias(d-bias)$; // update bias
9. packet.payload $\leftarrow sc$;
10. send(packet); // send the data

Figure 6: An SC table corresponding to the Huffman table and delta values

<table>
<thead>
<tr>
<th>Delta value</th>
<th>Huffman code</th>
<th>Shortest code $e_b=2, e_r=2$</th>
<th>bias</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>00010</td>
<td>0101</td>
<td>1</td>
</tr>
<tr>
<td>-3</td>
<td>0101</td>
<td>101</td>
<td>2</td>
</tr>
<tr>
<td>-2</td>
<td>0000</td>
<td>101</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>0101</td>
<td>101</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>100</td>
<td>101</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>111</td>
<td>111</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0011</td>
<td>111</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>0111</td>
<td>111</td>
<td>-2</td>
</tr>
<tr>
<td>4</td>
<td>00101</td>
<td>0111</td>
<td>-1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 7: Index addressing to keep the default SC table usable for successive approximation. Accumulated error is -2, and the error range is ±2.
In general, the shortest code for representing comparing these two entries is sufficient to find the shortest

\[ E_L \text{ and } E_H \]

In the previous example the SC table was constructed for the indexes \( E_L = -4 \) and \( E_H = +4 \), i.e.

\[ \left[ -4, 4 \right] \]

As shown in the table in Figure 8, the entry corresponding to \( d_2 \) contains the shortest code among delta values \( \left[ -4, 0 \right] \), i.e. \(-2 \pm 2\). Similarly, the entry corresponding to \( d_5 \) contains the shortest code among delta values \( [0, 4] \). As \( [-4, 0] \cup [0, 4] = [-4, 4] \), looking up and comparing these two entries is sufficient to find the shortest code in the desired range.

In general, the shortest code for representing \( d_i \) in the tolerable error range \( [E_L, E_H] \) can be found by comparing the set of entries in the SC table shown in Eq. (6).

\[
S = \left\{ d_i + E_L + j \times |e_L| + (j - 1) \times |e_H|, \\
1 \leq j \leq \left\lfloor \frac{E_H - E_L}{e_H - e_L} \right\rfloor \right\}
\]

In the previous example the SC table was constructed for \( e_L = -2 \) and \( e_H = +2 \). If the table shall be used for the new error range \( E_L = -4 \) and \( E_H = +4 \), then altogether \( \left\lfloor \frac{4 - (-4)}{2 - (-2)} \right\rfloor = 2 \) entries of the SC table need to be considered, \( E_L \text{ and } E_H \text{ to } +2 \text{ and } 2 \) need to be looked up. If in the case that the delta value is \( d_i = 0 \), the entries corresponding to \(-2 \pm 2\) need be used (see Figure 8).

Algorithm 2 describes how to support a wider range of acceptable error while the SC table is constructed for a shorter default error range. It shows that creating the SC table for a fixed default tolerable error range does not limit the system to that error range. In fact, the proposed method can also be used for any larger tolerable error range and it still benefits from the default SC table.

### 3.4 Table Size Reduction

The collected ECG signals of the same person can change from day to day, because of changes in person’s activity, conductivity of electrodes, muscle contraction, etc. Figure 9 shows the histogram of delta values corresponding to the ECG signal of one person for two different days measured on our ECG monitoring prototype (see Section 4.3). As shown in the figure, not only the occurrence frequency of delta value \(-1 \text{ to } -2\) is reduced by 4.7\%, but this delta value is not the most frequent value anymore. Changing occurrence frequencies of delta values can lead to a reduction in compression ratio, which demands updating the SC table (or generally the Huffman table). The SC table can be constructed (updated) either on the wearable device or on the Smartphone (server) and then transmitted to the wearable device. Due to the high resolution in recorded data, the range of ECG values, and consequently delta values, is large. It results in a large SC table (on average 1.47 MBits in our experiments). We use a simple but effective approach to reduce the size of the SC table (or generally Huffman table) while still retaining the same compression ratio.

Algorithm 2: Using shortest code table for a wider range or acceptable error

1. Inputs: Wider error range \( E_L \text{ and } E_H \)
2. \( Sp \leftarrow \text{getSample}(); \) // get the first signal’s sample
3. \( \text{bias} \leftarrow 0; \)
4. while \( \text{True} \) do
5. \( \text{Sn} \leftarrow \text{getSample}(); \) // new sample
6. \( d \leftarrow \text{Sn} - \text{Sp}; \) // delta value
7. \( d \leftarrow d \text{- bias}; \)
8. \( \text{Sp} \leftarrow \text{Sn}; \) // previous sample
9. \( \text{num_iter} \leftarrow \frac{E_H - E_L}{e_H - e_L}; \)
10. \( \text{min_val} \leftarrow \text{sc_table.get(d)}(); \)
11. for \( j \leftarrow 1 \) to \( \text{num_iter} \) do
12. \( d' \leftarrow d + E_H + j \times |e_L| + (j - 1) \times |e_H|; \)
13. \( \text{sc} \leftarrow \text{sc_table.get(d')}; \) // get shortest code
14. \( \text{bias} \leftarrow \text{bias_table.get(d');} \) // get bias
15. if \( \text{sc} \text{ is shorter than } \text{min_val} \) then
16. \( \text{bias} \leftarrow \text{bias' + d';} \)
17. \( \text{min_val} \leftarrow \text{sc} \text{;} \)
18. \( \text{packet.payload} \leftarrow \text{min_val}; \)
19. \( \text{send(packet);} \) // send the data
end

Although the delta values can theoretically vary from \(-2^{W-1}\) to \(+2^{W-1}-1\), in practice the appearance of large negative and large positive values is extremely rare, according to comprehensive observations. As shown in Figure 10, the delta values can be divided into two groups: 1) the rare range which includes large negative and large positive values that appear rarely, and 2) the usual range which includes the values that appear frequently. The delta values greater than a pre-defined threshold \( R_{th} \) or less than \(-R_{th} \) are considered as rare values.

It is worth noting that Huffman codes are proved [30] to have an upper bound that is shown in Eq. (7), where \( p_1 \) and \( p_2 \) are the probabilities of the least and second least frequent values, respectively, and \( n \) is the number of different values to be compressed (i.e. alphabets). According to the probability distribution of delta values, \( p_1 \) and \( p_2 \) are too small and tend to zero. Hence, the upper limit approaches

\[
\min \left\{ \left\lfloor \log_2 \left( \frac{\Phi + 1}{\Phi \times p_1 + p_2} \right) \right\rfloor, n - 1 \right\}, \quad \Phi = \frac{1 + \sqrt{5}}{2}
\]

In order to reduce the size of our SC table (or Huffman table), we only consider the usual range of delta values. For the rare range of values, we do not use Huffman coding, but store the raw \( W \) bits. We use a specific code word of the Huffman table as the delimiter to distinguish these two different ranges and codings. Figure 11 shows an example where some rare delta values appear. As these delta values do not have Huffman codes, they are presented in the
The main benefit of this approach is a significant reduction in the rare range appear extremely infrequent, and therefore coding approach will not necessarily increase or decrease the compression ratio significantly. In this hybrid approach, the delta values in the rare range are represented by only \( W \)-bit width codes. A delimiter is used to separate Huffman codes and raw data, such that the data can be decoded. In this hybrid approach, the delta values in the rare range are represented by only \( W \) bits instead of a long Huffman code, however, they need a delimiter to separate them from Huffman codes. It should be noted that this hybrid coding approach will not necessarily increase or decrease the compression ratio. This is due to the fact that delta values in the rare range appear extremely infrequent, and therefore they do not contribute to the compression ratio significantly. The main benefit of this approach is a significant reduction in table size as evaluated in Section 4.4.

4 Evaluation and Results

4.1 Experimental Setup

We used data from the MIT-BIH Long-Term ECG Database (ltdb) and MIT-BIH Arrhythmia Database (mitdb) [31] to evaluate our proposed technique. The Long-Term ECG Database provides 7 sets of two-channel ECG signals sampled at 128 Hz with 12-bit resolution for almost one day. The Arrhythmia database provides 48 sets of two-channel ECG signals sampled at 360 Hz with 11-bit resolution for half an hour. The first 40,000 samples of each ECG recording are used for training to construct the Huffman tables, while the rest are used for evaluating the compression rate of our techniques and state-of-the-art.

4.2 Comparison with state-of-the-art

Our proposed approximate compression approach for the Huffman technique can also be applied to other existing compression techniques, like [4, 6, 12, 14], to further improve their compression ratio. In other words, these techniques and our proposed approximate compression are complementary solutions.

To show the effectiveness of our proposed technique, we compare it with two state-of-the-art compression method, i.e. QLV [10] and [6]. Despite many other existing compression techniques, QLV [10] and our approximate compression are mutually exclusive. The reason is that QLV uses Huffman codes as separators to distinguish different levels, and therefore, it can not exploit approximation. Thus, we will compare our approach against QLV. Instead, [6] and our technique are orthogonal and can be used at the same time. We apply our approximate compression on top of the technique proposed in [6] to show the further compression ratio that can be achieved by our approximation technique.

To quantify the compression performance, we employ the compression ratio (CR) metric which is defined as

\[
CR = \frac{S_{\text{orig}} - S_{\text{comp}}}{S_{\text{orig}}} \times 100
\]

where \( S_{\text{orig}} \) and \( S_{\text{comp}} \) represent the size of original and compressed data, respectively. We consider an acceptable error range of up to ±4, which has a quite negligible impact on the quality of the reconstructed signal.

Figure 12a shows the compression ratio achieved using an exact Huffman technique and our approximate compression with different tolerable error ranges. As expected, the compression ratio is improved as the tolerable error range extends. Some ECG recordings, like ‘1’, ‘2’ and ‘4’, show a large potential for approximate compression. For some other recordings, like ‘3’, the compression ratio does not show large sensitivity to the tolerable error range (while still is beneficial). The reason behind this behavior is that the ECG recording ‘3’ belongs to a person whose heart beats are perfectly normal and regular. On the contrary, the ECG recording ‘1’ has some anomalies inducing irregularities in the data as a result of arrhythmia or unfiltered motion artifact [32]. Figure 14 shows a small number of ECG samples from these recordings. The irregularity and regularity of heart beats in those different recordings is easily visible.

The achieved compression ratio of our technique and QLV
(a) Compression ratio of our approximate compressor (for different acceptable error ranges) and exact Huffman compared to the uncompressed baseline

Figure 12: Compression ratio of our proposed approximate compressor compared to exact Huffman

(b) Compression ratio improvement of our approximate compressor (with ±4 error range) compared to the exact Huffman compression as baseline

Figure 13: Compression ratio of our compressor (with ±4 error range) compared to state-of-the-art QLV [10]

(a) Compression ratio of our approximate compressor and state-of-the-art [10] compared to the uncompressed baseline

(b) Compression ratio improvement of our approximate compressor compared to state-of-the-art [10] as baseline

Figure 14: ECG signals for two different recordings with regular and irregular rhythms

[10] are shown in Figure 13a. For all the ECG recordings, our approximate compressor outperforms the state-of-the-art approach [10]. For ECG recording ‘10’ the compression ratio of our technique is about 75%, while QLV [10] achieves only 33%. Figure 13b shows the compression ratio improvement of our technique compared to QLV [10]. Our approximate compressor achieves up to 60% data reduction (corresponding to ECG recording ‘10’) compared to QLV, i.e. the data volume to be transmitted is 60% smaller when using our approximate compression rather than using QLV.

Figure 15 shows the average number of bits to encode each ECG sample in QLV [10] and our approximate compression with an acceptable error range of ±4. The average number of bits per sample in QLV is always less than the raw data size (i.e. W = 12), but it is more than in our technique. It is worth noting that QLV uses Huffman codes as delimiters.

Figure 15: Average bits needed to store one ECG sample in our technique and QLV [10]

As the input data level changes frequently they must insert the delimiters often, which introduces an overhead that can erase their achieved compression. However, QLV does not need to keep a large Huffman table, which makes it suitable for devices in which memory is a scarce resource.

Figure 16 illustrates the compression ratio improvement achieved by applying our approximate compression on top of the frame-based compression presented in [6]. The effectiveness of [6] highly depends on the accuracy of the R-peak detection, which makes it more vulnerable to motion artifacts and noise.

4.3 Power Consumption & Lifetime Analysis

We implemented the techniques on a prototype version of our wearable ECG monitoring platform which is shown in Figure 17. For our processing unit, we consider a 32-bit ARM Cortex-M3 RISC embedded processor operating at a maximum speed of 96 MHz, which is designed for low power applications and meets the performance requirements for processing biomedical signals and data encryption (to ensure the security of transmitted data). A Bluetooth Low
Energy (BLE) modules is used to transmit the data to the smartphone. Although the power consumption of the BLE device is low with respect to classic Bluetooth or other similar devices, it is still the major contributor of the total energy consumption of the system. Table 2 shows the power consumption of the components including processor, Analog front end (AFE) and the Bluetooth (BLE) module in active mode and low-power mode.\textsuperscript{1}

To see the effect of our compression technique on the battery life, we considered a scenario and calculate the lifetime based on the power consumption values that are measured. In this scenario, to reduce the power consumption of the system, the components are duty cycled (i.e. they operate only for a very small time interval and spend the rest of the time in sleep mode)\textsuperscript{29}. Exploiting low power methods only for a very small time interval and spend the rest of the system, the components are duty cycled (i.e. they operate based on the power consumption values that are measured. In this scenario, to reduce the power consumption of the system, the components are duty cycled (i.e. they operate only for a very small time interval and spend the rest of the time in sleep mode)\textsuperscript{29}. Exploiting low power methods only for a very small time interval and spend the rest of the system, the components are duty cycled (i.e. they operate

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure16.png}
\caption{Compression ratio improvement of our approximate compressor applied on top of [6] compared to [6] as baseline}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure17.png}
\caption{ECG monitor prototype}
\end{figure}

\begin{table}[h]
\centering
\caption{Energy consumption of components}
\begin{tabular}{|c|c|c|}
\hline
Component & Active Mode & Low-power Mode \\
\hline
Processor @ 20MHz & 10.7 & 0.001 \\
Analog Front End & 0.17 & <0.0005 \\
Bluetooth module & 17.5 & 0.003 \\
\hline
\end{tabular}
\end{table}

\begin{table}[h]
\centering
\caption{Battery lifetime of different schemes}
\begin{tabular}{|c|c|c|}
\hline
Scheme & Avg. Current [mA] & Battery life [days] \\
\hline
No compression & 3.37 & 5 \\
QLV [10] & 2.84 & 7 \\
Ours & 1.78 & 10 \\
\hline
\end{tabular}
\end{table}

\begin{table}[h]
\centering
\caption{Measured values for parameters in Eq. (9)}
\begin{tabular}{|c|c|c|}
\hline
Table & SC\textsubscript{2} [KBits] & M\textsubscript{L} [bits] & N\textsubscript{D} \\
\hline
Full & 1669 & 1971 & 4096 \\
Usual & 472 & 981 & 1924 \\
\hline
\end{tabular}
\end{table}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure18.png}
\caption{The size of the SC table when considering the full range (both usual- and rare-range), and the usual range}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure19.png}
\caption{The size of the SC table when considering the full range (both usual- and rare-range), and the usual range}
\end{figure}

\textsuperscript{1}The current consumption of low-power modes, and active mode of processor are from datasheet. The other values are measured on our ECG monitor prototype and verified by datasheet.

\textsuperscript{2}The current consumption of low-power modes, and active mode of processor are from datasheet. The other values are measured on our ECG monitor prototype and verified by datasheet.

To see the effect of our compression technique on the battery life, we considered a scenario and calculate the lifetime based on the power consumption values that are measured. In this scenario, to reduce the power consumption of the system, the components are duty cycled (i.e. they operate only for a very small time interval and spend the rest of the time in sleep mode)\textsuperscript{29}. Exploiting low power methods only for a very small time interval and spend the rest of the system, the components are duty cycled (i.e. they operate based on the power consumption values that are measured. In this scenario, to reduce the power consumption of the system, the components are duty cycled (i.e. they operate only for a very small time interval and spend the rest of the time in sleep mode)\textsuperscript{29}. Exploiting low power methods only for a very small time interval and spend the rest of the system, the components are duty cycled (i.e. they operate based on the power consumption values that are measured. In this scenario, to reduce the power consumption of the system, the components are duty cycled (i.e. they operate

\textsuperscript{2}The current consumption of low-power modes, and active mode of processor are from datasheet. The other values are measured on our ECG monitor prototype and verified by datasheet.

\textsuperscript{2}The current consumption of low-power modes, and active mode of processor are from datasheet. The other values are measured on our ECG monitor prototype and verified by datasheet.

\textsuperscript{2}The current consumption of low-power modes, and active mode of processor are from datasheet. The other values are measured on our ECG monitor prototype and verified by datasheet.
(359 KBits on average). Reducing the table size makes our proposed approximate compressor able to be adapted and updated at runtime by creating an updated SC table using recent data. The updated SC table can be created either on the wearable device, or on a smartphone/server and then be transmitted to the wearable device.

5 Conclusions

This paper presented an approximate compression technique for biomedical signals (e.g. ECG) to reduce the energy consumption of data transmission in wearable healthcare monitoring systems. Our approximate compressor takes advantage of error tolerance in biomedical signals and finds the shortest Huffman code for each delta value in the user-specified acceptable error range. Then, we proposed a method to reduce the computational cost for finding the approximate result that is based on a pre-computed table and a mechanism to keep the accumulated error in the acceptable error range. This mechanism is capable to support a temporarily larger error range in cases where the quality of the reconstructed signal is not the main concern.

Experimental results show that our proposed techniques achieve more than 60% reduction in wireless communication (i.e. data size that is to be transmitted) compared to the state-of-the-art approach [10], and more than 40% improvement when applied on top of the state-of-the-art approach [6]. In addition, we reduced the size of the dictionary that is used for compression by 1 MBits, on average. Altogether, by using our approximate compressor, the battery lifetime of our ECG monitoring prototype extends from 7 to 10 days.

Acknowledgment

This work was partly supported by the German Research Foundation (DFG), Research Training Group GRK 1194. Authors would also like to thank Mr. Moises Araya for his help in setting up the evaluation platform.

References