




## Direct Lightweight Temporal Compression for Wearable Sensor Data

Lucie Klus<sup>1,2\*</sup> , Roman Klus<sup>1\*</sup>, Elena Simona Lohan<sup>1\*\*</sup> , Carlos Granell<sup>2</sup>, Jukka Talvitie<sup>1\*\*\*</sup> ,  
Mikko Valkama<sup>1\*\*</sup> , and Jari Nurmi<sup>1\*\*</sup> 

<sup>1</sup>Electrical Engineering Unit, Tampere University, 33014 Tampere, Finland

<sup>2</sup>Institute of New Imaging Technologies, Universitat Jaume I, 12071 Castellón de la Plana, Spain

\*Student Member, IEEE

\*\*Senior Member, IEEE

\*\*\*Member, IEEE

Manuscript received December 15, 2020; revised December 31, 2020; accepted January 11, 2021. Date of publication January 14, 2021; date of current version February 8, 2021.

**Abstract**—Emerging technologies enable massive deployment of wireless sensor networks across many industries. Internet of Things (IoT) devices are often deployed in critical infrastructure or health monitoring and require fast reaction time, reasonable accuracy, and high energy efficiency. In this letter, we introduce a lossy compression method for time-series data, named direct lightweight temporal compression (DLTC), enabling energy-efficient data transfer for power-restricted devices. Our method is based on the lightweight temporal compression method, targeting further reconstruction error minimization and complexity reduction. This letter highlights the key advantages of the proposed method and evaluates the method's performance on several sensor-based, time-series data types. We prove that DLTC outperforms the considered benchmark methods in compression efficiency at the same reconstruction error level.

**Index Terms**—Sensor signal processing, data compression, direct lightweight temporal compression (DLTC), Internet of Things (IoT), lightweight temporal compression (LTC), redundancy reduction, time series.

### I. INTRODUCTION

Internet of Things (IoT) has experienced dramatic growth in recent years, infiltrating industry, medical care, and many other areas in society. IoT devices range from simple sensors to complex devices used in critical applications, such as haptic gloves and autonomous cars. As the increasing volume of transferred data can limit the performance of critical applications, compression methods are gaining attention in the realm of massive deployment of wearable and IoT devices. The proposed compression method, named direct lightweight temporal compression (DLTC), has been developed for use in low-latency, sensor-based wearables, such as smart watches, activity trackers, and light-weight monitoring devices, which have limited computational resources due to hardware or battery limitations and delay sensitivity for proper functionality. The primary goal of our method is to reduce data size for fast transfer and more efficient storage, with the ability to reconstruct the measured entity as accurately as possible at any arbitrary time. The proposed algorithm is adjustable in means of compression ratio (CR), which directly correlates with the resulting compression error.

Recent work comparing various compression methods [1], including lightweight temporal compression (LTC), autoencoder methods, compressive sensing or discrete wavelet transform on biometric signals, has shown that LTC is highly efficient when applied to wearable sensor data. According to [1], LTC achieves top results when compressing photoplethysmogram (PPG) and respiration (RESP) signals, when low CRs are considered (e.g., CR up to 40 for RESP data). Similar conclusions were drawn in [2]. Therefore, based on the good performance of LTC reported in the literature so far, the considered methods utilized in this letter focus on LTC and its derivatives.

LTC was first proposed in [3] in 2004, showing compression with up to 20 : 1 size reduction with the reconstruction error significantly smaller than the specified metering device's accuracy. The complexity of the method is  $O(1)$ . The LTC method was presented as the efficient simplification of piece-wise linear approximation method, specifically designed for sensor-based, 1-D data.

Numerous extensions and improvements to the LTC were proposed including adaptive lightweight temporal compression (ALTC) [4], which modifies the LTC to sacrifice compression efficiency for an improved performance on noisy and dynamic data, or refined lightweight temporal compression (RLTC) [5], which extends the LTC technique with additional bins for higher compression efficiency, at the expense of an increased complexity. An extension of LTC to multiple dimensional data compression was presented in [6]. LTC and other compression schemes for the sensor-based data were also addressed in [7]–[9].

In this letter, we present a new LTC-based compression technique, DLTC, and compare it with other LTC variants from the literature. The main contributions are as follows.

- 1) We introduce DLTC: a novel, efficient, and simple compression technique for sensor-based, time-series data.
- 2) We discuss the improvements of the proposed method in comparison to the benchmark LTC, most importantly reduced latency and improved compression error.
- 3) We compare DLTC with the benchmark LTC-based methods and show that the proposed method outperforms the remaining ones by a significant margin.

The rest of this letter is organized as follows. Section II introduces the benchmark methods and their implementation, Section III presents the novel data compression method, followed by the utilized datasets, major improvements of the method, and future work in Section IV. Finally, Section V concludes this letter.

Corresponding author: Lucie Klus (e-mail: lucie.klus@tuni.fi).

Associate Editor: L. Azpilicueta.

Digital Object Identifier 10.1109/LENS.2021.3051809

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see <https://creativecommons.org/licenses/by/4.0/>

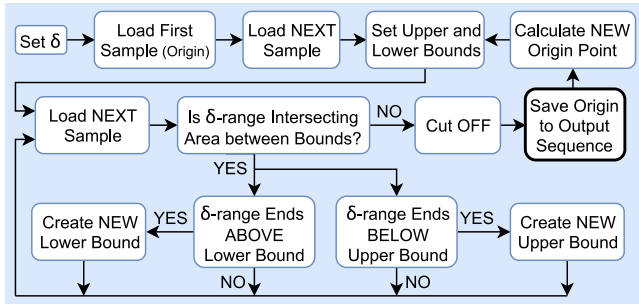


Fig. 1. Block diagram of the LTC algorithm.

## II. BENCHMARK LTC-BASED ALGORITHMS

This section presents the original LTC and the two benchmark LTC-based extensions: RLTC and ALCT. All of these methods, same as proposed DLTC, are designed for time-series, 1-D data such as seismic activity, atmospheric or blood pressure, electric power measurements, temperature, etc.

### A. Traditional LTC

LTC implementation in this letter is based on the original implementation proposed in [3]. The initial goal of this method is to compress 1-D sensor-based, environmental data prior to transmission while focusing on computational simplicity. Subsequent application of this method in a variety of letters [1], [4], [6], [8], [10] has determined the suitability of LTC for sensor-based data compression in general. LTC has a single parameter  $\delta$ , which is added and subtracted from  $y$ -axis coordinate of each sample to create a tolerance interval ( $\delta$ -range). The  $\delta$  parameter defines the tradeoff between the maximum error caused by the lossy compression and the achievable data size reduction. The algorithm is briefly described in the following paragraph and depicted in Fig. 1. The pseudocode and the guide for the LTC implementation can be found in [2]–[4], [6], and [10].

After defining the  $\delta$  parameter, the first sample's coordinates from the time-series we want to compress are set as the first origin point, and two lines called upper and lower bound are created. Upper bound intersects the origin point and the  $\delta$ -distance above the second sample. Similarly, lower bound line is created by intersecting the origin point and  $\delta$ -distance below the second sample. The following sample is loaded, and if its  $\delta$ -range is not within valid upper and lower bounds, cutoff is initiated. The current origin point is added to the output sequence. The new origin point is calculated, as a mean between upper and lower bounds at last in-bound sample's  $x$ -coordinate. The cutoff is finalized by creating new upper and lower bounds between the newly created origin point and the  $\delta$ -range of the out-of-bound's sample. If the new sample's  $\delta$ -range is found at least partially within bounds, the algorithm adjusts bounds according to Fig. 1. If the  $\delta$ -range ends above lower bound, the new lower bound is created by intersecting the valid origin point and the minimum of the  $\delta$ -range. The upper bound is checked similarly, after which the next sample is considered.

### B. Refined Lightweight Temporal Compression

The RLTC, introduced and evaluated in [5], further extends the idea of its predecessor. The approach and the algorithm enhance the traditional LTC by introducing multiple bounds to increase the search space of the method. The complexity of the method is increased from  $O(1)$  to  $O(n)$ , where  $n$  represents the number of considered bounds. In case  $n = 1$ , the method is equivalent to LTC. The letter compares the

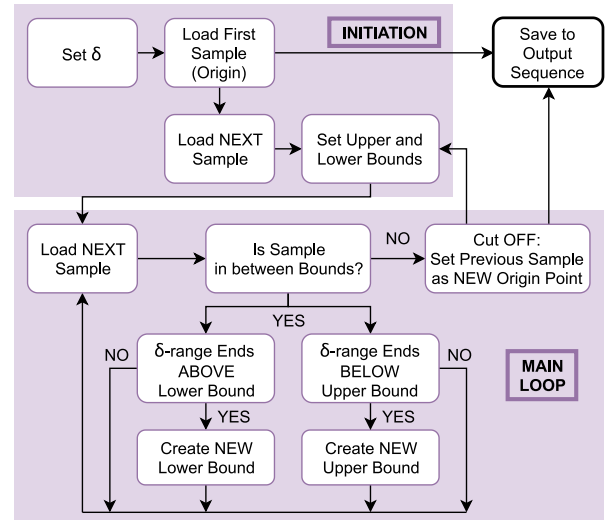


Fig. 2. Block diagram of the DLTC algorithm.

RLTC to LTC in terms of CRs under the same size of parameter  $\delta$ , and in terms of energy consumption in an IoT device under the assumption of periodic data transfer. We choose this method for comparison, as the creators claim it outperforms LTC in the terms listed above. The algorithm implementation is as presented in [5].

### C. Adaptive Lightweight Temporal Compression

ALTC, proposed in [4], adjusts the original method to better represent the noise within the data while keeping the complexity at  $O(1)$ . In the letter, the authors utilize lossless differential pulse code modulation to further reduce the size of the data, which is not considered in this work for fair evaluation of all considered methods. The authors adjust LTC in the cutoff phase, where instead of adding only the origin point to the output file, they add the origin as well as the current (out of bound) sample there. The new origin point is still created at the last valid sample's  $x$ -axis value in between the upper and lower bounds, and the new bounds are created using the current sample. The algorithm was recreated using the pseudocode presented in [4].

## III. NOVEL DLTC

The term “direct” in DLTC was chosen to express the straightforward approach of the proposed algorithm with complexity  $O(1)$ . The main objective of this method is to further reduce the number of operations by the algorithm, minimize the reconstruction error, and avoid the generation of the data points that were not present in the original set. The following paragraphs briefly describe the algorithm (see Fig. 2), followed by more thorough mathematical explanation (see Fig. 3) and then highlight the key differences with LTC.

After defining the  $\delta$  parameter, the first sample's coordinates are set as the first origin point and are added to the output sequence. Two lines called upper and lower bound are created. Upper bound intersects the origin point and the  $\delta$ -distance above the second sample. Similarly, lower bound line is created by intersecting the origin point and  $\delta$ -distance below the second sample. If the following sample is not within valid upper and lower bounds, cutoff is initiated. The last sample found within bounds is set as new origin point and is immediately added to the output sequence. The cutoff is finalized by creating new upper and lower bounds between the new origin point and the  $\delta$ -range of

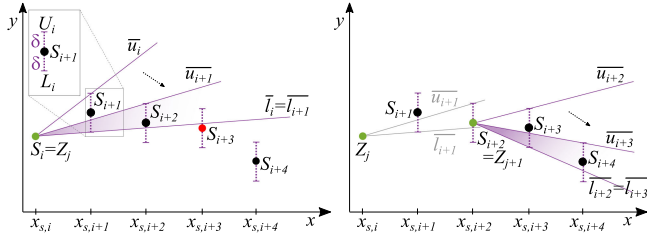


Fig. 3. DLTC algorithm visualization: First sample is set as first output sample  $Z_j$ . Sample  $S_{i+1}$  is used to create upper  $\bar{u}_i(\cdot)$  and lower  $\bar{l}_i(\cdot)$  bounds. Sample  $S_{i+2}$  decreases the upper bound to  $\bar{u}_{i+1}(\cdot)$ . Sample  $S_{i+3}$  is out of bounds (left). The new origin point  $Z_{j+1}$  is set, and the whole algorithm is initiated again starting with creating new bounds from  $Z_{j+1}$  (right).

the out-of-bound's sample. If the new sample is found within bounds, the algorithm adjusts bounds according to Fig. 2. If the  $\delta$ -range ends above lower bound, a new lower bound is created by intersecting the valid origin point and the minimum of the  $\delta$ -range. The upper bound is checked similarly, after which the next sample is considered.

Below, we define DLTC algorithm in mathematical expressions. Let  $s = \{S_1, \dots, S_n\}$  be the input sample sequence, where  $S_i = [x_{s,i}, y_{s,i}]$ ,  $i = 1, \dots, n$  is the  $i$ th input sample point, with  $x_{s,i}$  representing the sample index and  $y_{s,i}$  being the measurement value. Similarly, let  $z = \{Z_1, \dots, Z_m\}$  be the output sample sequence where  $Z_j = [x_{z,j}, y_{z,j}]$  is the  $j$ th output sample, where  $j = 1, \dots, m$  while  $m < n$ .

The algorithm is initiated by setting the first input sample  $S_i$  as  $Z_j$  ( $Z_1 \triangleq S_1$ , Fig. 3 left, green dot), which is also immediately added to the output set  $z$ . Next, the upper and lower bounds ( $\bar{u}_i(\cdot)$  and  $\bar{l}_i(\cdot)$ , respectively) are constructed by intersecting  $Z_j$  with  $U_i = [x_{s,i+1}, y_{s,i+1} + \delta]$  and  $L_i = [x_{s,i+1}, y_{s,i+1} - \delta]$ .

The rest of the algorithm is operated per-sample in iterative fashion as follows. Based on the next ( $S_{i+2}$ ) sample, adequate  $U$  and  $L$  points are created ( $U_{i+1} = [x_{U,i+1}, y_{U,i+1}] = [x_{s,i+2}, y_{s,i+2} + \delta]$  and  $L_{i+1} = [x_{L,i+1}, y_{L,i+1}] = [x_{s,i+2}, y_{s,i+2} - \delta]$ ).

If  $\bar{l}_i(x_{s,i+2}) \leq y_{s,i+2} \leq \bar{u}_i(x_{s,i+2})$ , the new bounds are created as follows. If  $y_{U,i+1} \leq \bar{u}_i(x_{s,i+2})$ , the new upper bound  $\bar{u}_{i+1}(\cdot)$  is created by intersecting  $Z_j$  and  $U_{i+1}$ . Otherwise, the upper bound remains unchanged. If  $y_{L,i+1} \geq \bar{l}_i(x_{s,i+2})$ , the new lower bound  $\bar{l}_{i+1}(\cdot)$  is created by intersecting  $Z_j$  and  $L_{i+1}$ . Otherwise, the lower bound remains unchanged.

Alternatively, if  $y_{s,i+2} < \bar{l}_i(x_{s,i+2})$  or  $y_{s,i+2} > \bar{u}_i(x_{s,i+2})$ , cutoff is initiated (Fig. 3 left, red dot representing sample  $S_{i+3}$ ). The new initial point  $Z_{j+1}$  is created as the last valid sample and is immediately added to the output set  $z$  (Fig. 3 right, green dot). New set of bounds is created by intersecting the  $Z_{j+1}$  and the adequate  $U$  and  $L$  points of the sample found out of bounds ( $U_{i+3}$  and  $L_{i+3}$  of sample  $S_{i+4}$ ).

As the purpose of LTC-based algorithms is to compress linear trends within the data, their reconstruction  $r = \{R_1, \dots, R_n\}$  is realized by pair-wise first-order polynomial interpolation of the compressed data  $z$  at every original sample instance  $x_{s,i}$ .

In the following, we highlight the main *differences* between DLTC and LTC.

- 1) The initial points  $Z$  are added to the output set  $z$  immediately after their assignment.
- 2) The cutoff is realized after the considered sample itself does not belong to the area between bounds, not considering the  $U$  and  $L$  points around it, as the effect of  $\delta$  parameter is already applied through the degree of upper and lower bounds (higher  $\delta$  corresponds to the wider bounds).

- 3) Initial points are assigned at the coordinates of the last sample within bounds instead of interpolating between bounds.

The summary of the key *advantages* of DLTC over the other LTC variants is listed as follows (based on the differences above).

- 1) Due to the difference 1), the compression method's *latency is significantly reduced*, as each sample is immediately evaluated and in case of  $Z$ -assignment, immediately transmitted. The mean latency reduction  $\Delta L$  can be estimated as  $\Delta L = CR \cdot \mu_{s,x}$ , where  $CR$  is the compression ratio of the method and  $\mu_{s,x}$  is the mean sampling interval of the input data  $s$ .
- 2) The proposed method does not duplicate the error margin at every sample [see difference 2)]; therefore, the error and the degree of compression are suppressed at the same  $\delta$  level compared to the other methods. Due to this fact, the *compression error is lower and more stable*.
- 3) As DLTC does not produce any artificial data points [all samples from output set of data  $z$  belong to the input set of data  $s$  as well, see difference 3)], DLTC can be utilized as an efficient compression method, as well as *data-adaptive redundancy reduction technique*.
- 4) The method reduces the number of algebraic operations at each iteration of the algorithm. As a result, DLTC further *reduces the required energy and memory usage* during compression.

## IV. DATA AND RESULTS

### A. Dataset Description and Evaluation Metrics

The PPG and atmospheric pressure data were chosen from *An Open Dataset for Human Activity Analysis using Smart Devices* [11] for evaluation. The smartwatch model was the LG Watch Urbane 2 for the collection of the utilized wearable data, including 91 337 PPG heart rate measurements and 14 900 atmospheric pressure measurements.

The experiments were evaluated in MATLAB software, version R2019a, on a laptop computer with Intel Core i7-8750H CPU and 32 GB RAM.

The error of compression is calculated as the percentage root-mean-squared error (RMSE) [%] =  $\sqrt{\frac{1}{N} \sum_{n=1}^N [y_{s,n} - y_{r,n}]^2} \cdot \frac{100}{\text{range}(y)}$ , where  $y_{s,n}$  and  $y_{r,n}$  are  $y$ -axis values of the original and reconstructed data and  $\text{range}(y)$  is the difference between the maximum and the minimum value of the original data  $y_{s,i}$ .

The compression ratio  $CR$  represents the ratio between the size of the original data to the size of the compressed data (both in bits) as  $CR = \text{size}(s)/\text{size}(z)$ .

### B. Measurement-Based Results

The tradeoff achieved between the compression error and the  $CR$  between the considered methods is shown in Figs. 4 and 5 for PPG and atmospheric pressure data, respectively. Fig. 4 shows the comparable results between DLTC and RLTC methods for  $CR$ s below 10. At higher  $CR$ s, however, DLTC outperforms the other methods by a significant margin. Fig. 4 also shows the stability of the DLTC method in terms of error, as RMSE increases smoothly. Fig. 5 shows the results on the atmospheric pressure data. Here, DLTC also outperforms the other methods if the  $CR$  values are greater than 10. Both figures show the suitability of the proposed DLTC method, as it provides 33 and 81% higher compression than the original LTC at 2 and 3% RMSE error level, respectively.

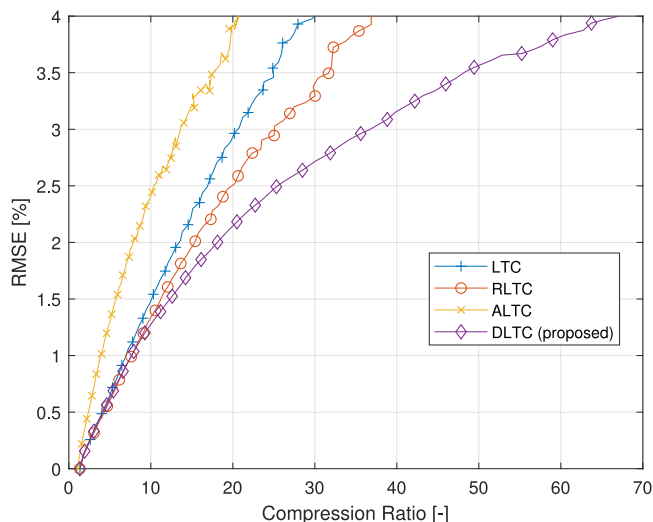


Fig. 4. Evaluation of RMSE based on CR for different methods, PPG data. The achieved CR at 2% RMSE was 13.62 for LTC, 15.42 for RLTC, 7.26 for ALTC, and 18.13 for the proposed DLTC.

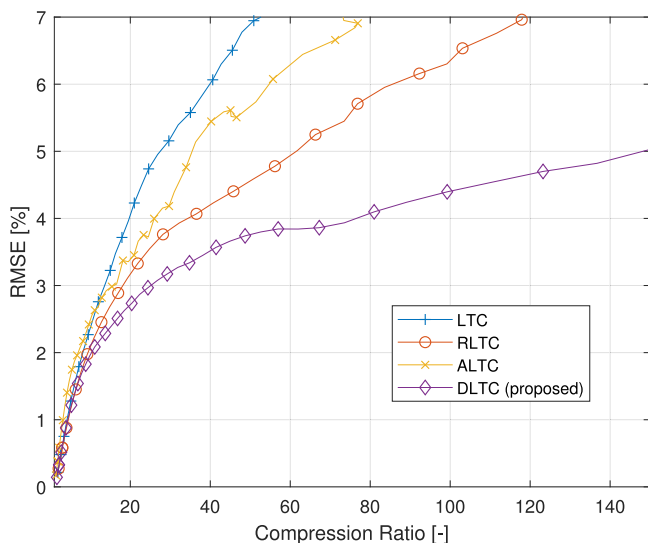


Fig. 5. Evaluation of RMSE based on CR for different methods, atmospheric pressure data. The achieved CR at 3% RMSE was 13.50 for LTC, 19.27 for RLTC, 16.58 for ALTC, and 24.40 for the proposed DLTC.

### C. Discussion and Open Issues

Based on the presented results, we have found that the DLTC method significantly outperforms the other considered LTC-related methods. The downside of the proposed DLTC method is the same data-dependence that happens in the original LTC algorithm. To further verify the proposed method's performance, the authors will perform additional validation on a larger number of data sets and more types of data. Future work will also include additional compression methods suitable for latency-sensitive time-series data. The focus will also be on the practical implementation of the method in real-world IoT scenarios, which will also allow measuring the energy efficiency of the proposed method directly to verify its complexity. Automatic, data-dependent  $\delta$ -parameter selection can also be addressed in the future, as currently it is either determined experimentally, or set as maximum tolerable error determined, e.g., by sensor's tolerance of measurements. Another limitation of DLTC, as for the original LTC, is its limitation to 1-D

temporal data. Extending the method to multidimensional space while minimizing the computational complexity is our additional future objective.

## V. CONCLUSION

In this letter, we introduce an efficient way to compress time-series data with minimum additional latency and reconstruction error compared to traditional LTC compression methods. We propose the DLTC algorithm, a novel lossy compression method derived from the well-known LTC. The main improvements include the minimization of the reconstruction error, lower amount of required operations, and redundancy reduction capabilities. We comparatively evaluated the performance of LTC, RLTC, ALTC, and the proposed DLTC on two types of sensor-based data, namely PPG and atmospheric pressure datasets. On both datasets, DLTC outperformed all other methods since it minimized compression error at the given CRs.

## VI. ACKNOWLEDGMENT

This work was supported in part by A-WEAR - EU's Horizon 2020 Marie Skłodowska Curie Grant Agreement 813278 and in part by the Academy of Finland under Grant 323244 and Grant 319994.

## REFERENCES

- [1] M. Hooshmand, D. Zordan, D. Del Testa, E. Grisan, and M. Rossi, "Boosting the battery life of wearables for health monitoring through the compression of biosignals," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1647–1662, Oct. 2017.
- [2] L. Klus, E. S. Lohan, C. Granell, and J. Nurmi, "Lossy compression methods for performance-restricted wearable devices," in *Proc. Int. Conf. Localization GNSS WiP*, 2020. [Online]. Available: <http://ceur-ws.org/Vol-2626/>
- [3] T. Schoellhammer, B. Greenstein, E. Osterweil, M. Wimbrow, and D. Estrin, "Lightweight temporal compression of microclimate datasets [wireless sensor networks]," in *Proc. 29th Annu. IEEE Int. Conf. Local Comput. Netw.*, 2004, pp. 516–524.
- [4] J. Azar, A. Makhoul, R. Darazi, J. Demerjian, and R. Couturier, "On the performance of resource-aware compression techniques for vital signs data in wireless body sensor networks," in *Proc. IEEE Middle East North Africa Commun. Conf.*, 2018, pp. 1–6.
- [5] O. Sarbishei, "Refined lightweight temporal compression for energy-efficient sensor data streaming," in *Proc. IEEE 5th World Forum Internet Things*, 2019, pp. 550–553.
- [6] B. Li, O. Sarbishei, H. Nourani, and T. Glatard, "A multi-dimensional extension of the lightweight temporal compression method," in *Proc. IEEE Int. Conf. Big Data*, 2018, pp. 2918–2923.
- [7] S. Vadrevu and M. S. Manikandan, "A new quality-aware quality-control data compression framework for power reduction in IoT and smartphone PPG monitoring devices," *IEEE Sensors Lett.*, vol. 3, no. 7, Jul. 2019, Art. no. 6001404.
- [8] M. J. Rubin, M. B. Wakin, and T. Camp, "Lossy compression for wireless seismic data acquisition," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 1, pp. 236–252, Jan. 2016.
- [9] G. Giorgi, "A combined approach for real-time data compression in wireless body sensor networks," *IEEE Sensors J.*, vol. 17, no. 18, pp. 6129–6135, Sep. 2017.
- [10] R. Sharma, "A data compression application for wireless sensor networks using LTC algorithm," in *Proc. IEEE Int. Conf. Electro/Inf. Technol.*, 2015, pp. 598–604.
- [11] S. Faye, N. Louveton, S. Jafarnejad, R. Kryvchenko, and T. Engel, "Open dataset for human activity analysis using smart devices," 2017. [Online]. Available: <http://www.sfaye.com/open-dataset-human-activity-analysis-using-smart-devices/>