Workshop Proceedings of the 19th International Conference on Intelligent Environments (IE2023)
G. Bekaroo et al. (Eds.)

doi:10.3233/AISE230005

Platform and Hardware Independent Reliable Signal Processing

Lukas P.A. ARTS [a,1], Joren PARIDAENS [b] and Egon L. VAN DEN BROEK [a]
[a] *Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands*
[b] *Coöperatie 9de online U.A., Utrecht, The Netherlands*
ORCiD ID: Lukas P.A. Arts https://orcid.org/0000-0001-8398-0259, Egon L. van den Broek https://orcid.org/0000-0002-2017-0141

Abstract. The recent explosion of sensors enable our environment to act in an intelligent way. These Intelligent Environments rely on sense making of the sensors' data streams. This process starts with reliable signal processing in real-time. This is challenging due to i) the low energy and computing resources of edge devices, ii) the signals' non-stationary nature, and iii) the variety in software and hardware. To tackle this triplet of challenges, we present a WebAssembly-based hardware and software independent fast Continuous Wavelet Transform (fCWT), which excels in processing non-stationary signals at low costs. The application shows to be 2x-5.5x faster than competitors on speech, electrocardiogram (ECG), and vibration signals, enabling reliable real-time processing on edge devices. This yields new opportunities for the creation of safe and reliable Intelligent Environments.

Keywords. intelligent environments, reliability, internet of things, edge computing, continuous wavelet transform, sensors

1. Introduction

The proliferation of Internet of Things (IoT) devices has led to an explosion of sensors that enable us to monitor and analyze our environment in new ways [1,2,3]. Analyzing the data streams generated by these devices can help us better understand and adapt to our surroundings and realize Intelligent Environments (IE) [4,5,6]. However, IE is challenging [7], starting with the real-time analysis of the sensors' signals that are often noisy and non-stationary (see Figure 1) [2,8,9]. Besides, analysis generally happens on edge computing devices where energy, network bandwidth, and computational resources are scarce [1]. Also, effective analysis is often troubled by the diversity of software and hardware standards IoT and IE knows [2].

A major reason why non-stationary frequency analysis is challenging, is the inability of traditional analysis techniques, such as Fourier analysis, to capture the temporal variations in non-stationary signals [10,11]. These static techniques ignore time and average, blur, or spread out the time-varying patterns (see Figure 2) [12,13]. Time-frequency algorithms overcome this limitation by analyzing the time-varying signals in both the

[1] Corresponding Author: l.p.a.arts@uu.nl

Figure 1. Non-stationary signals are omnipresent within IoT. Across six domains, non-stationary signal are used to increase reliability, safety and security in real-world applications.

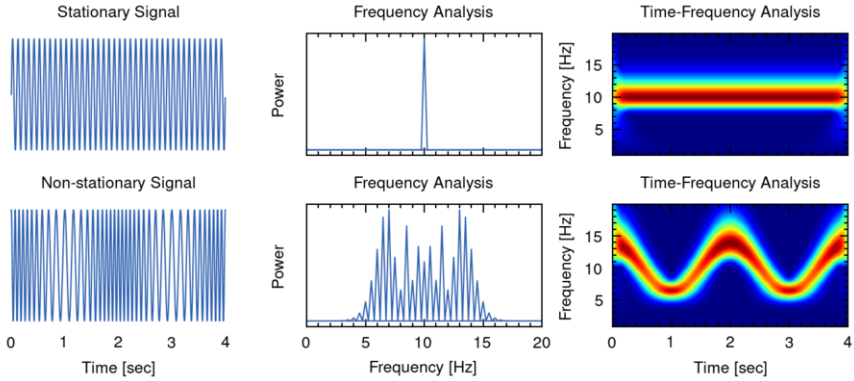

Figure 2. The analysis of stationary and non-stationary signals differs considerable. Analysing a non-stationary signal only in the frequency domain produces a blurred and spread out frequency spectrum. Instead, a time-frequency analysis is needed to reveal the time-dependent frequency components.

time and frequency domain [14]. However, analyzing signals in two dimensions simultaneously is computationally expensive, making real-time signal analysis on low-power IoT devices unfeasible [15,16]. Less expensive time-frequency techniques exist, but they achieve this speedup by lowering accuracy [17]. Hence, one is always forced to choose between speed or precision. This can be problematic in situations where both speed and accuracy are crucial. Without fast and efficient computation, cardiac health assessment and seizure prediction are too late in emergency situations [18], IoT devices cannot provide real-time feedback, and machine inspection by edge computing devices requires too much battery power [16]. Simultaneously, without high precision, safety, reliability, and security are no longer guaranteed. Minor but important changes in heart rhythm [19] or machine vibration [4,20] would go unnoticed, speech analysis fails [21], and security becomes an issue in radio frequency-based authentication of IoT devices [22].

Portability and flexibility of these non-stationary analysis techniques also limits use in edge computing applications [2]. As implementations are often hardware-specific, they are inflexible and challenging to adapt across multiple devices. Consequently, the lack of cross-platform and cross-device compatibility undermines their use in the diverse hardware landscape that IoT is rich [2,1]. Non-stationary analysis remains underdevel-

oped in comparison to traditional frequency techniques, resulting in untapped potential from real-time data streams generated by IoT devices [4].

To address the issue of speed, we developed a fast implementation of the high-resolution Continuous Wavelet Transform (CWT) algorithm was called fCWT [23]. During benchmarks it was shown to provide high-resolution frequency analysis at real-time speeds, essentially solving the speed-accuracy dilemma. However, its portability is limited. fCWT uses many hardware-dependent optimization strategies to speed up calculation. As such, fCWT is bounded to specific operating systems and CPU types, limiting its use in IoT. Implementing fCWT in a hardware-independent, portable language such as Python or Javascript is not an option as these languages are unable to implement fCWT's low-level optimization strategies destroying its much needed speed and efficiency advantage [16]. Until recently, this issue was unsolvable and low-level optimized code needed to be tuned for every hardware configuration. This all changed when WebAssembly [24] was introduced in 2017 (see 'WebAssembly: Why is it special').

WebAssembly presents a new hardware independent abstraction layer over modern hardware. Suddenly, programs could be hardware- and device-independent while still having the ability to access low-level instructions and achieve performances comparable to native implementations. WebAssembly has already been coined the next big thing in edge computing offloading [25,26]. Since its release, several attempts have been made to bring the power of time-frequency algorithms, such as the Short-Term Fourier Transform (STFT) [17], Discrete Wavelet Transform (DWT) [14], and CWT [11], via WebAssembly to the edge. However, despite efforts, current WebAssembly-based time-frequency implementations still lack the high resolution and noise-resilience of CWT [27], are not fully optimized [28], or lack a publicly available source code [29]. As a result, the power of WebAssembly's native performance and portability has not yet been fully utilized for non-stationary signal analysis in resource-limited edge computing devices to make our environments safer, secure, or more reliable [16,30].

This paper presents WebfCWT, a WebAssembly implementation of fCWT. WebfCWT inherits fCWT's efficient calculation while being much more portable. To show this, we benchmarked WebfCWT on multiple signals and created a publicly accessible web application, enabling researchers to perform non-stationary signal analysis in a user-friendly and secure browser environment. Since WebfCWT runs entirely client-side, all data remains on the user's system, ensuring privacy by design [25]. Furthermore, WebfCWT can be integrated seamlessly into IoT and edge computing devices as it inherits WebAssembly's portability [26,31].

Next, we provide i) a recap of the mathematical theory behind CWT, which enables non-stationary signal analysis, ii) fast Continuous Wavelet Transform (fCWT)'s Fourier-based implementation and iii) the compilation of fCWT to WebAssembly. Section 3 presents a benchmark of WebfCWT's speed compared to three other implementations. Section 4 ends this paper with a discussion.

WebAssembly: Why is it special?
Arguably, Python, Javascript, and other portable languages already offered the portability that WebAssembly [24] promises. What, exactly, is the advantage of WebAssembly over these widely used languages? The major thing is that these languages offer portability by limiting access to low-level hardware instructions such as Single Instruction Multiple Data (SIMD) and custom memory management.

Ahead-of-time (AOT) languages such as C, C++ and Rust, that compile target-specific binaries, do provide these optimizations but can only do this by sacrificing portability. For different Central Processing Unit (CPU)'s, a C, C++ or Rust codebase needs to be recompiled, or even rewritten, in severe cases. Thread handling using OpenMP, for example, does not run natively on every system. WebAssembly brings together the best of both worlds. With WebAssembly one can harness the power of low-level programming languages while still being portable [25]. Consequently, computationally intensive tasks such as non-stationary signal processing, can now be performed by small, portable WebAssembly programs that run on any edge computing device no matter their operating system or hardware configuration [31].

2. Methodology

WebfCWT is a WebAssembly-based program that utilizes the fast CWT to offer a cross-platform and hardware-independent signal analysis. CWT is capable of tracking the frequency components responsible for a signal's non-stationary behavior over time by breaking the input signal down into wavelets, which are localized in both time and frequency. Using WebAssembly, the majority of fCWT's optimization strategies can be made hardware-independent, without compromising performance. To show the power of fast, hardware-independent, non-stationary signal analysis, a publicly available browser application [32] was developed that embeds fCWT directly in the browser making it free and accessible for everyone.

2.1. The Continuous Wavelet Transform

Wavelets are defined as a-scaled and b-shifted versions of a base wavelet

$$\psi_{ab}(t) = \psi\left(\frac{t-b}{a}\right). \tag{1}$$

Analogues to the sine and cosine waves in Fourier Transform (FT) [11,10,14], a linear combination of wavelets shifted across time (b) and frequency (a) can represent any signal by

$$f(t) = \sum_{a,b=-\infty}^{\infty} c_{ab}\psi_{ab}(t), \tag{2}$$

where each wavelet is scaled by c_{ab}, the wavelet coefficient. Like FT, we can calculate c_{ab} for all a and b by calculating the convolution between the signal $f(t)$ and the conjugated wavelet family $\overline{\psi}_{ab}(t)$

$$c_{ab} = W_\psi f(a,b) = |a|^{-1}\int_{-\infty}^{\infty} f(t)\overline{\psi}_{ab}(t)\mathrm{d}t, \tag{3}$$

which is the Wavelet Transform (WT)of $f(t)$ using wavelet $\psi(t)$. Calculating the WT at dyadic scale-factors $a = 2^i$ and offset-parameters $b = 2^i k, k \in N$ obtains the definition

of the Discrete WT (DWT) that uses orthogonal wavelets to minimize the amount of wavelets needed to reconstruct a signal without loss [33] . Sampling a and b continuously obtains a time-frequency estimation often used when in-depth signal analysis is preferred. Such a WT is called a Continuous WT (CWT) [11,34]. When Eq. 3 is implemented digitally, the continuous integral translates to a discrete summation

$$W_\psi f[a,b] = |a|^{-1} \sum_{n=0}^{N-1} f[n] \overline{\psi} \left[\frac{n-b}{a} \right], \tag{4}$$

which is mathematically equivalent to passing the input signal through a series of band-pass filters with different cutoff frequencies.

2.2. The fast Continuous Wavelet Transform

We can improve CWT's efficiency when applying Parseval's Theorem to Eq. 3:

$$c_{ab} = W_\psi f(a,b) = \frac{1}{2\pi} \int \hat{f}(\xi) \overline{\widehat{\psi_{ab}}(\xi)} \mathrm{d}\xi, \tag{5}$$

where $\hat{f}$ and $\widehat{\psi_{ab}}$ are the Fourier-transformed input and wavelet, respectively. Next, we substitute the scaled and shifted Fourier-transformed wavelet $\widehat{\psi_{ab}}(\xi)$ by its mother wavelet:

$$c_{ab} = W_\psi f(a,b) = \frac{1}{2\pi} \int \hat{f}(\xi) \overline{\hat{\psi}(a\xi)} e^{ib\xi} \mathrm{d}\xi, \tag{6}$$

which provides us with the inverse FT of $\hat{f}(\xi)\overline{\hat{\psi}(a\xi)}$. Subsequently, fCWT uses a variety of optimization strategies in combination with already existing Fast Fourier Transform (FFT) libraries [35] to implement these three steps as efficient as possible. For details on this implementation, we refer to [23].

2.3. WebfCWT

The introduction of WebAssembly in 2017 represented a significant milestone in the development of high-performance web-based edge computing [24]. WebAssembly introduced a new abstraction layer providing applications access to low-level operations such as Single Instruction Multiple Data (SIMD) [36] and custom memory management to achieve near-native performances while still allowing them to run across different platforms and devices. We compiled fCWT's C++ based codebase and all its dependencies (i.e., FFT libary [35]) to a multi-threaded, vectorized WebAssembly environment using Emscripten, webworkers, and the experimental SIMD instructionset to bring fast, hardware-efficient and hardware-independent, non-stationary signal analysis to IE [25,26,31].

To show the advantages of having a fast, accurate, and hardware-independent CWT implementation, we developed a publicly available web application implementing fCWT [23] using WebAssembly [32]. The application is designed to provide a user-friendly environment that allows users to interact with CWT without requiring program-

ming knowledge or experience in the digital signal processing domain. Moreover, the application is designed with accessibility and safety in mind, allowing users to execute CWT exclusively client-side. As data never leaves the user's system, that analysis of privacy-sensitive data is ensured to be safe.

3. Results

To evaluate the speed of WebfCWT, we conducted a comparative analysis of its execution times with other implementations of the CWT. WebfCWT was tested against CCWT.js [28], the only existing open source WebAssembly-based CWT, and Matlab's Wavelet toolbox [37] to show the difference between WebfCWT's real-time, hardware-independent performance and that of a large software suite that uses all available hardware.

In order to validate the efficacy and versatility of WebfCWT, we selected three distinct real-world, non-stationary signals often used in edge computing to increase safety, reliability, and security. Specifically, a self-recorded audio fragment containing speech was chosen to evaluate performance in the context of broadband audio analysis. Hence, a wide frequency range was chosen with a low number of frequencies. High-resolution analysis speed was assessed using 5000 frequencies on a short Electrocardiogram (ECG) segment of record 109 from the MIT-BIH database [38] and real-time, multi-stream analysis on the first four channels of NASA's roller bearing dataset record '2003.10.22.13.09.13'. [39]. Number of frequencies in the speech and vibration tests were considerably lower due to the large sample frequencies and the 4Gb memory constraint of WebAssembly.

Table 1. Three real-world, non-stationary signals were used in the comparative analysis between WebfCWT and two competitors. Signals were analysed on different frequency ranges at different resolutions.

Signal Type	Sample Frequency	Freq. Range	Num. of Freq.	Duration	Goal
Speech	16000Hz	1-8000Hz	300	6s	Broadband
ECG	360Hz	1-20Hz	5000	50s	High-resolution
Vibration	20000Hz	1-10000Hz	500	4x1s	Multi-channel

The majority of the energy of the ECG signal is in the 1-20Hz frequency band [40]. Hence, only those frequencies were analysed. In contrast, the entire frequency range of the speech and vibration signal were analysed as consonants (i.e., "s", "h", and "f") are in the high-frequency range [41] and vibrations often occur at high frequencies equal to the machine's rotational speed [20]. The benchmark was performed on an eight-core 2.30-GHz CPU running macOS Big Sur. Details about each analysis are listed in Table 1. Benchmark results are shown in Figure 3.

4. Discussion

In the previous section, we have seen that WebfCWT outperforms all competitors, by being at least twice as fast. When compared to CCWT.js, the only other open source WebAssembly-based CWT implementation, WebfCWT is even 4-5.5 times faster. As such, no real alternative for WebfCWT currently exists, making it a valuable addition

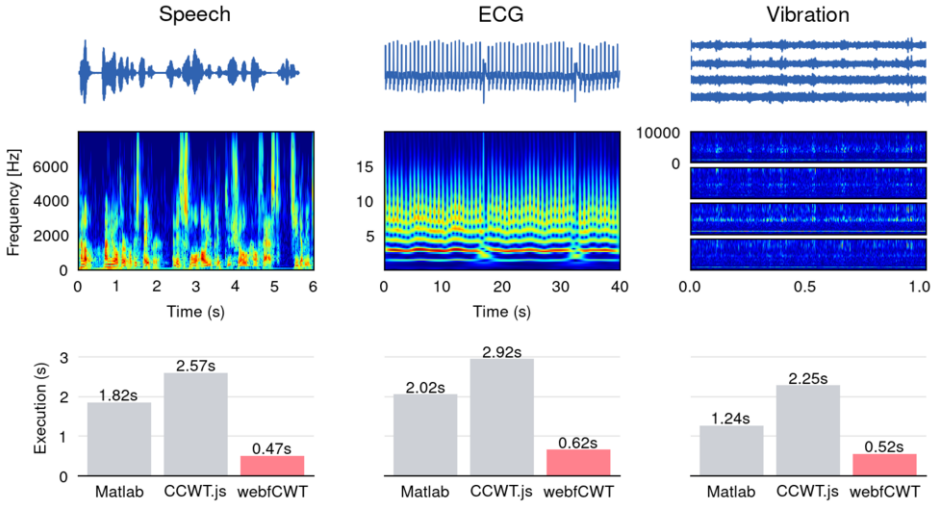

Figure 3. Benchmark between WebfCWT, its WebAssembly competitor CCWT.js, and Matlab on three signals commonly encountered in edge computing and IoT solutions. Speech, heart rate and vibration. Details about each analysis are shown in Table 1. Execution times are averaged on 10 runs.

to the fast but stationary FFT, limited STFT, and already existing CWT's. WebfCWT showed to perform real-time non-stationary signal analysis while being software and hardware independent.

WebfCWT's benchmark results highlight the effectiveness of several low-level optimization strategies that would not be possible without WebAssembly. For example, on the high-frequency vibration dataset, WebfCWT's horizontal parallelization over the inverse FFT, SIMD usage in the point-wise multiplication, and pre-allocation of memory used by the time-frequency matrix made it possible to achieve a real-time performance where this was not possible before.

At present, the web application [32] is still limited to a visualization tool. However, it has been designed to function as a versatile platform for CWT-based analysis of various signals. In the future, signal-specific analysis modules could be implemented to provide researchers with tools such as phoneme recognition or noise removal for speech analysis [21], peak detection [42], filtering or heart rate variability extraction [43] for ECG research, diagnostic reporting on vibration data [20], or ridge extraction [44] for general purposes. Furthermore, other CWT-based time-frequency algorithms such as the Synchrosqueezed Transform (SST) [45] or Superlet Transform (SLT) [46] could be implemented to improve accuracy even more.

WebfCWT has no limitations beyond those inherited from the CWT, such as reduced frequency resolution at high frequencies [47], and those inherited from WebAssembly, such as the 4Gb memory limit. The benchmark has the limitation that it did not extensively test WebAssembly's claim of being hardware-independent. Future research could investigate the use of WebfCWT on various platforms, including edge devices and a range of other sensors [4], to determine if its performance in terms of speed, accuracy, and power consumption remains consistent. Nevertheless, current results are promising [5,6].

The proliferation of the IE has led to an explosion of sensors that generate non-stationary signals. Consequently, the analysis of these signals is essential for developing safe and reliable IE. Time-frequency analysis algorithms are designed to capture the temporal variations present in these signals; but, their computational complexity has made them challenging to implement in real-time and to use on different platforms. To address these challenges, we developed WebfCWT, a fast, high-resolution, platform and hardware-independent edge application of the fCWT, using WebAssembly technology. WebfCWT can provide high-quality, hardware independent frequency analysis at at least twice the speed of hardware dependent implementations. As such, it opens up new opportunities for the development of applications that make our environments safer, more reliable and secure and highlights the potential of WebAssembly for implementing complex algorithms in a hardware-independent and accessible way.

Acknowledgements

L.P.A.A. and E.L.v.d.B. have received funding for this work from the European Union's Horizon 2020 research and innovation program under grant agreement No. 952095, the IM-TWIN project. The funders had no role in study design, data collection and analysis, decision to publish or preparation of the manuscript.

References

[1] Zhang T, Gao L, He C, Zhang M, Krishnamachari B, Avestimehr AS. Federated learning for the internet of things: applications, challenges, and opportunities. IEEE Internet of Things Magazine. 2022;5(1):24-9.
[2] Jamshed MA, Ali K, Abbasi QH, Imran MA, Ur-Rehman M. Challenges, applications and future of wireless sensors in Internet of Things: A review. IEEE Sensors Journal. 2022.
[3] Esenogho E, Djouani K, Kurien AM. Integrating artificial intelligence Internet of Things and 5G for next-generation smartgrid: A survey of trends challenges and prospect. IEEE Access. 2022;10:4794-831.
[4] Lu S, Lu J, An K, Wang X, He Q. Edge Computing on IoT for Machine Signal Processing and Fault Diagnosis: A Review. IEEE Internet of Things Journal. 2023.
[5] Lee JH, Hashimoto H. Intelligent spaceconcept and contents. Advanced Robotics. 2002;16(3):265-80.
[6] Hall DL, Steinberg A. Dirty secrets in multisensor data fusion. Pennsylvania State Univ University Park Applied Research Lab; 2001.
[7] Augusto JC, Callaghan V, Cook D, Kameas A, Satoh I. Intelligent environments: a manifesto. Human-centric Computing and Information Sciences. 2013;3(1):1-18.
[8] Fahy C, Yang S, Gongora M. Scarcity of labels in non-stationary data streams: A survey. ACM Computing Surveys (CSUR). 2022;55(2):1-39.
[9] van den Broek EL. Monitoring technology: The 21st century's pursuit of well-being? Bilbao, Spain: European Agency for Safety and Health at Work (EUOSHA); 2017.
[10] Boashash B. Time-frequency signal analysis and processing: A comprehensive reference. 2nd ed. London, UK: Academic Press / Elsevier Ltd.; 2016.
[11] Boukouvala E, Miridakis N, Veloni A. Digital and statistical signal processing. CRC Press; 2019.
[12] Fano RM. Transmission of Information: A statistical theory of communications. Cambridge, MA, USA: The M.I.T. Press; 1961.
[13] de Souza UB, Escola JPL, da Cunha Brito L. A survey on Hilbert-Huang transform: Evolution, challenges and solutions. Digital Signal Processing. 2022;120:103292.
[14] Cohen L. Time-frequency analysis. Upper Saddle River, NJ, USA: Prentice Hall PTR; 1995.

[15] Karvat G, Schneider A, Alyahyay M, Steenbergen F, Tangermann M, Diester I. Real-time detection of neural oscillation bursts allows behaviourally relevant neurofeedback. Communications Biology. 2020;2:#72.
[16] Capra M, Peloso R, Masera G, Ruo Roch M, Martina M. Edge computing: A survey on the hardware requirements in the internet of things world. Future Internet. 2019;11(4):100.
[17] Stockwell RG, Mansinha L, Lowe RP. Localization of the complex spectrum: The *S* transform. IEEE Transactions on Signal Processing. 1996;44(4):998-1001.
[18] Aledhari M, Razzak R, Qolomany B, Al-Fuqaha A, Saeed F. Biomedical IoT: enabling technologies, architectural elements, challenges, and future directions. IEEE Access. 2022;10:31306-39.
[19] Cacioppo JT, Tassinary LG, Berntson GG. Handbook of Psychophysiology. 4th ed. Cambridge, UK: Cambridge University Press; 2017.
[20] Mukherjee A, Parlikad AK, McFarlane D. Ubiquitous Domain Adaptation at the Edge for Vibration-based Machine Status Monitoring. IEEE Internet of Things Journal. 2022.
[21] Vani H, Anusuya M. Improving speech recognition using bionic wavelet features. AIMS Electronics and Electrical Engineering. 2020;4(2):200-15.
[22] Ren Z, Ren P, Zhang T. Deep RF device fingerprinting by semi-supervised learning with meta pseudo time-frequency labels. In: 2022 IEEE Wireless Communications and Networking Conference (WCNC). IEEE; 2022. p. 2369-74.
[23] Arts LPA, van den Broek EL. The fast continuous wavelet transformation (fCWT) for real-time, high-quality, noise-resistant time–frequency analysis. Nature Computational Science. 2022;2(1):47-58.
[24] Haas A, Rossberg A, Schuff DL, Titzer BL, Holman M, Gohman D, et al. Bringing the web up to speed with WebAssembly. In: Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation; 2017. p. 185-200.
[25] Hoque MN, Harras KA. WebAssembly for Edge Computing: Potential and Challenges. IEEE Communications Standards Magazine. 2022;6(4):68-73.
[26] Wallentowitz S, Kersting B, Dumitriu DM. Potential of WebAssembly for Embedded Systems. In: 2022 11th Mediterranean Conference on Embedded Computing (MECO). IEEE; 2022. p. 1-4.
[27] Six J. pffft.wasm: an FFT library for the web; 2022. Available from: `https://github.com/JorenSix/pffft.wasm[LastaccessedonFebruary28,2023]`.
[28] Meißner A. WebSpectrogram; 2020. Available from: `https://github.com/Lichtso/WebSpectrogram[LastaccessedonFebruary28,2023]`.
[29] Hussain R. Wavelets Online; 2020. Available from: `https://rafat.github.io/wavelib/[LastaccessedonFebruary28,2023]`.
[30] Basir R, Qaisar S, Ali M, Aldwairi M, Ashraf MI, Mahmood A, et al. Fog computing enabling industrial internet of things: State-of-the-art and research challenges. Sensors. 2019;19(21):4807.
[31] Vaño R, Lacalle I, Sowiński P, Palau CE, et al. Cloud-Native Workload Orchestration at the Edge: A Deployment Review and Future Directions. Sensors. 2023;23(4):2215.
[32] Arts LPA. The fast Continuous Wavelet Transform Online; 2023. Available from: https://fcwt.app.
[33] Dremin IM, Ivanov OV, Nechitailo VA. Wavelets and their uses. Physics-Uspekhi. 2001;44(5):447.
[34] Addison PS. Introduction to redundancy rules: The Continuous Wavelet Transform comes of age. Philosophical Transactions of the Royal Society A: Mathematical, Physical, and Engineering Sciences. 2018;376(2126):#20170258.
[35] Frigo M, Johnson SG. FFTW: An adaptive software architecture for the FFT. In: Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181). vol. 3. IEEE; 1998. p. 1381-4.
[36] Chakrabarti C, Vishwanath M. Efficient realizations of the Discrete and Continuous Wavelet Transforms: From single chip implementations to mappings on SIMD array computers. IEEE Transactions on Signal Processing. 1995;43(3):759-71.
[37] Misiti M, Misiti Y, Oppenheim G, Poggi JM. Wavelet toolbox. The MathWorks Inc, Natick, MA. 1996;15:21.
[38] Moody GB, Mark RG. The impact of the MIT-BIH arrhythmia database. IEEE Engineering in Medicine and Biology Magazine. 2001;20(3):45-50.
[39] Lee J, Qiu H, Yu G, Lin J, Rexnord Technical Services. Bearing Data Set. NASA Ames Prognostics Data Repository. 2007. Available from: http://ti.arc.nasa.gov/project/prognostic-data-repository.
[40] Elgendi M, Jonkman M, DeBoer F. Frequency bands effects on QRS detection. Pan. 2010;5(15):1-5.

[41] Näätänen R, Lehtokoski A, Lennes M, Cheour M, Huotilainen M, Iivonen A, et al. Language-specific phoneme representations revealed by electric and magnetic brain responses. Nature. 1997;385(6615):432-4.
[42] Yochum M, Renaud C, Jacquir S. Automatic detection of P, QRS and T patterns in 12 leads ECG signal based on CWT. Biomedical signal processing and control. 2016;25:46-52.
[43] Cartas-Rosado R, Becerra-Luna B, Martínez-Memije R, Infante-Vazquez O, Lerma C, Perez-Grovas H, et al. Continuous wavelet transform based processing for estimating the power spectrum content of heart rate variability during hemodiafiltration. Biomedical Signal Processing and Control. 2020;62:102031.
[44] Iatsenko D, McClintock PV, Stefanovska A. Extraction of instantaneous frequencies from ridges in time–frequency representations of signals. Signal Processing. 2016;125:290-303.
[45] Daubechies I, Lu J, Wu HT. Synchrosqueezed wavelet transforms: An Empirical Mode Decomposition-like tool. Applied and Computational Harmonic Analysis. 2011;30(2):243-61.
[46] Moca VV, Bârzan H, Nagy-Dăbâcan A, Murean RC. Time-frequency super-resolution with superlets. Nature communications. 2021;12(1):337.
[47] Strang G. Wavelets. American Scientist. 1994;82(3):250-5.