

Time-Frequency Analysis for Feature Extraction Using Spiking Neural Network

Moshe Bensimon*, Yakir Hadad*, Yehuda Ben-Shimol, *Member, IEEE*, and Shlomo Greenberg, *Member, IEEE*

Abstract—Time-frequency analysis plays a crucial role in various fields, including signal processing and feature extraction. In this article, we propose an alternative and innovative method for time-frequency analysis using a biologically inspired spiking neural network (SNN), encompassing both specific spike-continuous-time-neuron (SCTN) based neural architecture and an adaptive learning rule. We aim to efficiently detect frequencies embedded in a given signal for the purpose of feature extraction. To achieve this, we suggest using an SN-based network functioning as a resonator for the detection of specific frequencies. We developed a modified supervised Spike-Timing-Dependent Plasticity (STDP) learning rule to effectively adjust the network parameters. Unlike traditional methods for time-frequency analysis, our approach obviates the need for segmenting the signal into several frames, resulting in a streamlined and more effective frequency analysis process. Simulation results demonstrate the efficiency of the proposed method, showcasing its ability to detect frequencies and generate a Spikegram akin to the Fast Fourier Transform (FFT) based spectrogram. The proposed approach is applied to analyzing EEG signals demonstrating an accurate correlation to the equivalent FFT transform.

Index Terms—Neuromorphic computing, time-frequency analysis, SNN, frequency detection, STDP, features extraction

I. INTRODUCTION

NEUROMORPHIC computing aims to substitute traditional methods with algorithms inspired by the human brain. Recent advancements in neuromorphic computing have shown that Spiking Neural Networks (SNNs) can achieve substantially reduced power consumption compared to conventional deep learning methods [1] [2].

The extraction of features in the frequency domain is a common approach for various applications [3] [4] [5]. Usually, this involves dividing the signal into overlapping segments (i.e. frames) and extracting features from each segment individually. However, this approach introduces several drawbacks, including computation delays, redundant calculations, and limited resolution [6] [7]. Adapting time-continuous methods for feature extraction overcame this limitation, allowing a more efficient and accurate time signal analysis.

In recent years, researchers are actively seeking alternative methods of applying time-frequency analysis for feature extraction purposes [8] [9] [10]. M. Adeli et al. [11] use a

cochleagram feature extraction method, which incorporates a bio-inspired model for time-frequency processing.

In a recent study, we proposed the use of an SCTN-based phase-shifting circuit for frequency detection and extracting MFCC-like features in acoustic applications [12]. This paper proposes an innovative method for time-frequency analysis using a biologically inspired spiking neural network (SNN). We suggest using an SCTN-based network for feature extraction in the frequency domain and functioning as a resonator circuit to detect specific frequencies. A modified supervised STDP learning approach is presented to train the network and optimize the weights for targeted frequency detection. Simulation results demonstrate the ability of the proposed method to efficiently extract frequency features. The proposed approach is applied to analyzing and classifying EEG signals and serves as a low-cost and efficient alternative to the well-known FFT transformation for feature extraction.

II. SCTN-BASED ARCHITECTURE FOR FREQUENCY ANALYSIS

A. SCTN-Spike Continuous Time Neuron Model

This section briefly describe the SCTN model, presented in [2]. The SCTN model is given by Equation (1).

$$Vm_j(t) = Vm_j(t-1) + \sum_{i=1}^N W_{ij} \cdot I_i(t) - \mu_j \quad (1)$$

where, the membrane potential, $Vm(t)$, is given by:

$$Vm(t) = \begin{cases} Vm_0, & t = 0 \\ (1-2^{-LF})(Vm(t-1) + \sum_{j=1}^N (w_j \cdot I_j + \phi) + \Theta), & t \cdot \text{mod}(LP) = 0 \\ Vm(t-1) + \sum_{j=1}^N w_j \cdot I_j + \phi + \Theta, & t \cdot \text{mod}(LP) \neq 0 \end{cases} \quad (2)$$

The parameters, Leakage Factor (LF) and Leakage Period (LP), represent the neuron's integrator leakage rate and the integrator operation rate, respectively. The parameters Θ and ϕ signify the pre-leakage and post-leakage biases, respectively. Every SCTN cell encompasses a leaky integrator whose time constant is governed by a specific parameter: $\alpha = 1 - 2^{-LF}$. This enables the SCTN to delay incoming signals based on a projected leakage time constant. For an appropriate choice of the LP and LF parameters, the SCTN has the ability to generate a phase shift of $\frac{\pi}{4}$ for a given frequency f_0 [12]:

$$f_0 = \frac{f_{Pulses} \cdot (1 - \alpha)}{2\pi \cdot (1 + LP)} = \frac{f_{Pulses}}{2^{LF} \cdot 2\pi \cdot (1 + LP)} \quad (3)$$

(Corresponding author: Shlomo Greenberg)

M. Bensimon, Y. Hadad, Y. Ben-Shimol, and S. Greenberg are with the School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer-Sheva, 84105, Israel. (e-mail: bensimmo@post.bgu.ac.il, hadadyak@post.bgu.ac.il, benshimo@bgu.ac.il, and shlomog@bgu.ac.il). S. Greenberg is with the Computer Science Department, Sami Shamoon College of Engineering, Beer-Sheva, 84100, Israel. (e-mail: shlomgr@sce.ac.il)

* M. Bensimon, and Y. Hadad contribute equally.

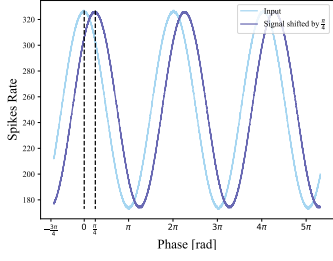
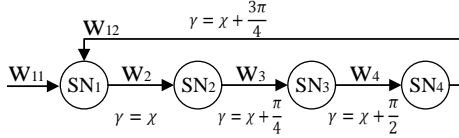
Fig. 1: A phase shift of $\frac{\pi}{4}$ by using single SCTN neuron.

Fig. 2: SNN-based Resonator architecture.

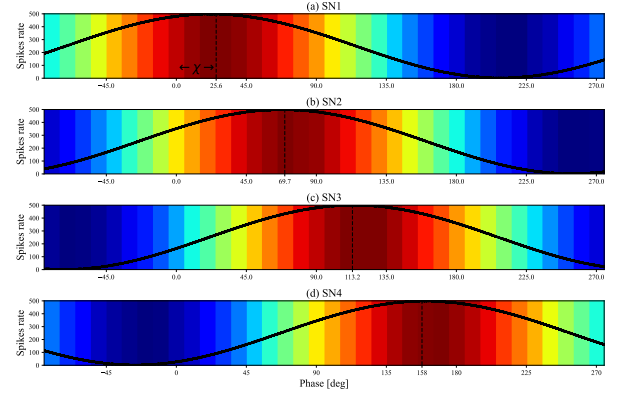
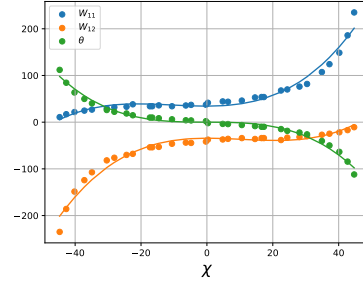
B. SNN-based architecture for frequency detection

This section presents the SCTN-based resonator circuit which serves as a frequency detector utilizing the SCTN basic phase shifting feature. Leveraging the fundamental phase-shifting feature of SCTN, we suggest using SNN architecture for detecting the embedded frequency components inherent in a given signal.

Fig. 2 shows the SCTN-based network that operates as a resonator circuit. The proposed circuit comprises four sequentially linked SCTNs. The output of the fourth neuron (SN_4) is connected to the first neuron representing a shift of $\chi + \frac{3\pi}{4}$ for the feedback path. Each neuron is designed to perform a phase shift of $\frac{\pi}{4}$, while the phase shift at the output of the first neuron (χ) ranges from 0 to $\frac{\pi}{4}$. The first neuron SN_1 has actually two input synapses: (a) the input signal, and (b) a signal shifted by $\chi + \frac{3\pi}{4}$. Fig. 3 depicts the four SCTN neurons' outputs demonstrating a shift of $\frac{\pi}{4}$ by each neuron. Setting the phase shift of the first neuron as χ allows an additional degree of freedom in designing and determining the bandwidth of the filter ($f_0 \pm \Delta f_0$). Fig. 4 depicts the synapse weights w_{ij} and bias Θ_1 as a function of the shift phase χ of the first neuron (SN_1). The superposition of the two input synapses to SN_1 yields the desired shift χ ($0 - \frac{\pi}{4}$). The resonator circuit has the capability to detect a broad spectrum of frequencies embedded in a time-series signal utilizing the SCTN basic phase shifting feature. The LP and LF parameters can be configured to detect any desired frequency, represented by the resonance frequency, f_0 . Upon detecting frequencies in the range of $f_0 \pm \Delta f_0$ the output neuron (SN_4) fire a train of pulses, where the maximum rate achieved at f_0 .

III. A MODIFIED STDP-BASED SUPERVISED LEARNING

This section provides a detailed description of the training process of the network aimed at detecting a desired frequency (f_0). First, we need to determine the LF and LP parameters of each SCTN using Eq. 3. Then, the SNN training phase is carried out using a modified STDP learning rule for adjusting

Fig. 3: The output of the four SCTN neurons: (a) shift of $\chi(25^\circ)$ by SN_1 , (b-d) shift of $\frac{\pi}{4}$ by SN_2 , SN_3 , SN_4 . Each bin reflects the number of spikes within a time window $N = 500$ time stamps.Fig. 4: The synapses weights (w_{ij}) and bias (Θ_i) as a function of the phase shift χ .

and determining the W_{ij} and Θ_i of the four neurons, to achieve the desired frequency response.

The loss function is defined as the Mean Squared Error (MSE) between each SCTN neuron output and its desired output. This MSE error should be minimized to ensure a phase shift of $\frac{\pi}{4} \pm \Delta\xi$, and a gain of $g_i^{ref} \pm \Delta g$. For simulation purpose we generate four predefined desired signals (one per each SCTN neuron output) to be used in the supervised training phase. The loss function is formalized as follows:

$$Loss = \frac{1}{4} \sum_{i=1}^4 MSE(SN_i^{out}(\xi_i, g_i), V_i^{ref}(\xi_i^r, g_i^{ref})) \quad (4)$$

where the SCTN output (SN_i^{out}) is given by:

$$SN_i^{out} = g_i \cdot \sin(2\pi\xi_i t), \quad i = 1, 2, 3, 4 \quad (5)$$

and the desired output (V_i^{ref}) is defined as follows:

$$V_i^{ref} = g_i^{ref} \cdot \sin(2\pi\xi_i^{ref} t + \chi + \frac{\pi}{4} \cdot (i-1)), \quad i = 1, 2, 3, 4 \quad (6)$$

where ξ_i , g_i , and ξ_i^{ref} , g_i^{ref} are the actual and desired phase and gain respectively. χ represents the shifting phase of SN_1 .

A. SNN training algorithm

This section presents the learning algorithm used to train the SNN model. To determine the values of the weight and

TABLE I: XOR function applied to the supervised STDP

SN_i^{out}	V_i^{ref}	$A \oplus B$
0	0	$0 \rightarrow \emptyset$
0	1	$1 \rightarrow STDP$
1	0	$1 \rightarrow Anti-STDP$
1	1	$0 \rightarrow \emptyset$

biases, we employ a modified STDP learning algorithm. Unlike the common unsupervised STDP method [13], we suggest applying a supervised learning approach. This means that the updating of the network weights is not triggered only when the SCTN fires a pulse, but also according to the reference desired signal.

Below is a description of the learning process. First, we randomly determine W_{ij} and Θ_i and the values of the neuron's synapses. In the first step, the goal is to change the Θ_i values in order to produce at each SCTN output, an average pulse rate of $\frac{N}{2}$ pulses per cycle consisting of windows of N time samples per window. This actually represents the DC of the signal and allows the stretching of the amplitude in the entire pulse range (0 to N). To obtain the desired pulse rate, we change the Θ_i values, during the learning process, using an inverse-Sigmoid-like function as follows:

$$\Delta\Theta = \left(\frac{2x - N}{N}\right)^2 \cdot \text{sign}\left(\frac{N}{2} - x\right) \quad (7)$$

where x is the current pulse rate. After the Θ_i has been tuned and the desired average pulse rate has been achieved, the learning process continues in order to determine the network weights through supervised STDP learning.

During the learning process, the weights of each SCTN neuron are updated according to the following STDP rule [13]:

$$\Delta W_{i,j} = \sum_{k \in S_{pre}} \sum_{i \in S_{post}} \begin{cases} A \cdot e^{-|t_i - t_k|/\tau}, & \text{if } t_i > t_k \\ -A \cdot e^{-|t_i - t_k|/\tau}, & \text{if } t_i \leq t_k \end{cases} \quad (8)$$

As stated before, the updating of the network weights is not triggered only when the SCTN fires a pulse, but also according to the reference desired signal. To compare the desired signal to the SCTN pulse- rate the reference signal should be encoded in a pulse-like signal (like PDM). Fig. 3 shows the encoding of the signal phase into spikes representation (y-axis), where each timestamp is determined by the number of spikes within a given window of size N . The desired spikes are randomly distributed within this window and used as a reference for the STDP learning rule. To decide whether or not to update the SNN weights, an XOR operation is performed between the SCTN output and the reference signal, as depicted in Table I, strengthening or weakening of the synapse weight is carried out according to the STDP rules given by Eq. 8. The parameters A and τ represent the learning rate and the time constant respectively. Choosing of a relatively small learning rate A ensures robust learning [14]. The time constant τ (usually on the order of several millisecond) defines the time interval used to control the weight changes. The learning rate A may depend on the difference between the actual and the desired amplitude, and is defined by:

$$A = (1 + \rho) \cdot 10^{-4} \quad (9)$$

where:

$$\rho = \frac{|g_i^{ref} - g_i|}{g_i^{ref}} \quad (10)$$

The time constant τ may be adjusted according to the absolute value of the phase error. A large error enables a larger time interval for controlling the required weight changes for convergence, and vice versa. Therefore, τ is defined as a function of the phase error given by:

$$\tau = K \cdot \frac{|\xi_i^{ref} - \xi_i|}{2\pi} \quad (11)$$

where, K is determined as a function of the sample frequency.

B. Avoiding local minima

The initial values of the Θ_i (as described in Sec. III-A) may lead to convergence to a local minima. In practice, during the learning process there may be a scenario where the phase converges while the amplitude error remains constantly large. To avoid this, we suggest to continue simultaneously updating both the bias Θ_i and the weights W_{ij} , according to the following equations:

$$\Theta_i = \Theta_i \pm \Delta\psi, \quad g_i^{ref} - g_i \geq \Delta\xi \quad (12)$$

Consequently, we modify the weights based on the spike rate:

$$W_{ij} = W_{ij} \mp 2\Delta\psi, \quad g_i^{ref} - g_i \geq \Delta g \quad (13)$$

where $\Delta\psi$ is constant value set to 0.002. Using these rules preserves the desired average pulse rate ($\frac{N}{2}$ pulses per cycle), enabling stretching of the amplitude. Simulations show that the improved learning process optimizes the learning speed and avoids convergence to local minima.

C. Convergence of the learning process

The convergence learning process is depicted in Fig. 5. The goal is to converge to the desired signal such that the amplitude extends over a defined range and each neuron represents its appropriate phase ($\chi, \chi + \frac{\pi}{4}, \chi + \frac{\pi}{2}, \chi + \frac{3\pi}{4}$). The x-axis represents the phase and the y-axis the magnitude encoded according to the pulse rate (sort of PDM). The current and desired signals of each neuron are marked with solid and dashed lines respectively. The desired phase range ($\xi_i^{ref} \pm \Delta\xi$) for each neuron are indicated by colored longitudinal bars. The stopping condition is occurred when all neurons met the following two criteria:

- (a) The phase converges to the desired value: $\xi_i^{ref} \pm \Delta\xi$.
 - (b) The magnitude converges to a desired range: $g_i^{ref} \pm \Delta g$.
- The amplitude values are represented as a function of the desired pulse rate in a given time window (N). Fig. 6 depicts the convergence of the MSE, amplitude, and phase errors during the learning process while applying the above-suggested learning rate and time constant. A minimal error is achieved after about 650 epochs for both the phase and the amplitude. Fig. 6c shows that after about 100 epochs the phase converges to the desired value while the amplitude error is still high. Moreover, the MSE for four of the SCTN neurons (especially for SN_4) is still high. Therefore, the suggested

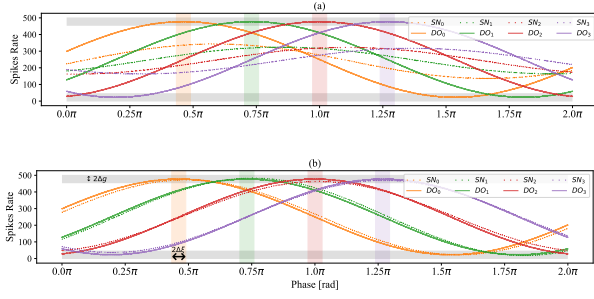


Fig. 5: The learning process: The neuron output for (a) random weight values, and (b) convergence of the network to the desired signal (for $\Delta g = 15$ and $\Delta \xi = 3^\circ$).

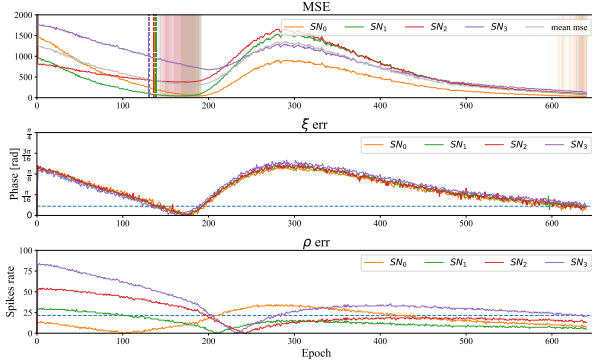


Fig. 6: Error convergence during the learning process: (a) MSE, (b) amplitude, and (c) phase error. The horizontal dashed lines indicates the desired convergence range.

improved learning process is triggered and the bias Θ_i and weights W_{ij} are updated according to Equations 12, and 13.

As a result, although at first, it is possible to notice an increase in the error, after an additional short training round (about 100 epochs) the error, with respect to both phase and amplitude, decreases for all neurons, and the MSE smoothly converges to zero after about 650 epochs. This illustrates the efficiency of the proposed STDP model enabling a smooth convergence to the desired phase. The learning process converges after 1000 epochs on average, where each training round contains a complete cycle of the signal with 40 sampling windows and 500 samples in each window (a total of 20K samples for a complete cycle of the signal).

IV. EXPERIENTIAL RESULT

To evaluate the efficiency of the proposed approach we applied the SNN-based resonator to detect the embedded frequencies in EEG signals. The EEG signal is encoded using pulse density modulation (PDM). The LF and LP parameters of the four SCTN neurons, which compose the resonator, are determined according to Eq. 3 to represent specific EEG frequency within the delta range (0.1 - 4Hz). Then, the SNN is trained to detect the desired frequency adapting the weights (W_{ij}) and the biases (Θ_i) for each SCTN neuron. The EEG signal is composed of 5 sub-bands: Delta (0.1-4 Hz), Theta (4-8 Hz), Alpha (8-14 Hz), Beta (14-32 Hz),

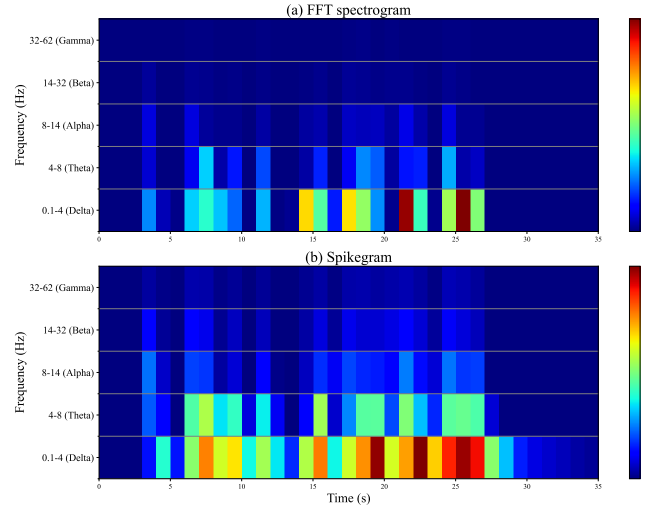


Fig. 7: EEG frequency features: (a) FFT spectrogram, and (b) SNN-based Spikegram.

and Gamma (32-60 Hz). To cover the whole frequency range each of the five EEG sub-bands is further divided into five specific frequencies. Therefore, an array of 25 SNN-based resonators (five for each sub-band) is required to detect the entire EEG frequency components. The frequency outputs of the resonators are represented using the Spikegram method as depicted in Fig. 7b demonstrating the occurrence of spikes over time for sub-band. The strength of the signal is shown in color where red represents the maximum spike rate. Fig. 7a depicts the spectrogram of the FFT of the same EEG signal under-test for the five sub-bands. A cross-correlation between the frequency features in the FFT domain and the frequency features extracted by the proposed SNN-based resonators show a high correlation between the two features set, ranging from 0.8 to 0.93. This is a very interesting result leading to the ability to use the proposed SNN-based approach as a low-cost and efficient alternative to the well-known FFT transformation.

To evaluate the quality of the SCTN-based features we utilize them for classifying situation awareness into three categories [15] using SNN classifier with unsupervised STDP learning. Results show a success rate of 96.8% for a 15-second EEG frame.

V. CONCLUSION

This article proposes a biologically inspired SNN-based resonator for accurate frequency detection and extracting features in the frequency domain. We present a new supervised STDP learning approach demonstrating a fast and efficient convergence. The quality of the extracted features has been examined for classifying EEG signals, demonstrating high classification accuracy. A comparison to frequency features in the FFT domain shows a high correlation with the SNN-based extracted features, suggesting the potential of applying this approach in various applications.

REFERENCES

- [1] C. D. Schuman, S. R. Kulkarni, M. Parsa, J. P. Mitchell, P. Date, and B. Kay, "Opportunities for neuromorphic computing algorithms and applications," *Nature Computational Science*, vol. 2, no. 1, pp. 10–19, 2022.
- [2] M. Bensimon, S. Greenberg, Y. Ben-Shimol, and M. Haiut, "A new sctn digital low power spiking neuron," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 8, pp. 2937–2941, 2021.
- [3] A. O. Salau and S. Jain, "Feature extraction: a survey of the types, techniques, applications," in *2019 international conference on signal processing and communication (ICSC)*. IEEE, 2019, pp. 158–164.
- [4] Y. D. Mistry, G. K. Birajdar, and A. M. Khodke, "Time-frequency visual representation and texture features for audio applications: a comprehensive review, recent trends, and challenges," *Multimedia Tools and Applications*, pp. 1–35, 2023.
- [5] P. Kataria, T. Sharma, and Y. Narayan, "A review of time, frequency and hybrid domain features in pattern recognition techniques," *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2022, Volume 2*, pp. 411–422, 2022.
- [6] Y. Yang, Z. Peng, W. Zhang, and G. Meng, "Parameterised time-frequency analysis methods and their engineering applications: A review of recent advances," *Mechanical Systems and Signal Processing*, vol. 119, pp. 182–221, 2019.
- [7] W. Qing-Hua, W. Li-Na, and X. Song, "Classification of eeg signals based on time-frequency analysis and spiking neural network," in *2020 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*. IEEE, 2020, pp. 1–5.
- [8] J. López-Randulfe, T. Duswald, Z. Bing, and A. Knoll, "Spiking neural network for fourier transform and object detection for automotive radar," *Frontiers in Neurobotics*, vol. 15, p. 688344, 2021.
- [9] E. Sejdić, I. Orović, and S. Stanković, "Compressive sensing meets time–frequency: An overview of recent advances in time–frequency processing of sparse signals," *Digital signal processing*, vol. 77, pp. 22–35, 2018.
- [10] N. Laurent and S. Meignen, "A novel time-frequency technique for mode retrieval based on linear chirp approximation," *IEEE Signal Processing Letters*, vol. 27, pp. 935–939, 2020.
- [11] M. Adeli, J. Rouat, S. Wood, S. Molotchnikoff, and E. Plourde, "A flexible bio-inspired hierarchical model for analyzing musical timbre," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 875–889, 2016.
- [12] M. Bensimon, S. Greenberg, and M. Haiut, "Using a low-power spiking continuous time neuron (sctn) for sound signal processing," *Sensors*, vol. 21, no. 4, p. 1065, 2021.
- [13] S. Song, K. D. Miller, and L. F. Abbott, "Competitive hebbian learning through spike-timing-dependent synaptic plasticity," *Nature neuroscience*, vol. 3, no. 9, pp. 919–926, 2000.
- [14] J. C. Thiele, O. Bichler, and A. Dupret, "Event-based, timescale invariant unsupervised online deep learning with stdp," *Frontiers in computational neuroscience*, vol. 12, p. 46, 2018.
- [15] Ç. İ. Acı, M. Kaya, and Y. Mishchenko, "Distinguishing mental attention states of humans via an eeg-based passive bci using machine learning methods," *Expert Systems with Applications*, vol. 134, pp. 153–166, 2019.