2024 IEEE Biomedical Circuits and Systems Conference (BioCAS) | 979-8-3503-5495-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/BioCAS61083.2024.10798280

Multiplierless Spiking Neural Network for Motor Signal Decoding in the Peripheral Nervous System

Qiaosong Deng, Junyu Ma, Hanfeng Cai, Hao You, Mustafa Kanchwala, Jianxiong Xu, Amirali Amirsoleimani, José Zariffa, and Roman Genov
Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada
jianxiong.xu@mail.utoronto.ca, roman@eecg.utoronto.ca

***Abstract*— The peripheral nervous system (PNS) facilitates communication between the brain and various organs. Advanced PNS neural interfaces can help in restoring motor functions for patients suffering from spinal cord injuries, amputations, and other conditions. The efficacy of these neural interfaces, and the precision of sensory activity decoding is heavily reliant on artificial intelligence techniques at the backend. Traditional machine learning techniques, such as feature-based classification, generally requires extensive feature engineering and domain expertise, and is often ineffective at handling very high-dimensional data. Additionally, convolutional neural networks (CNNs) tend to perform best when employing floating-point operations, which in-turn can be computationally intensive and less energy-efficient. Addressing these challenges, this paper presents a multiplier-less spiking neural network (SNN) that utilizes fewer neurons to achieve higher accuracy. Leveraging the discrete firing characteristics of SNNs, we propose a hardware implementation model, trained on experimentally recorded action potentials from rat peripheral nerves. This demonstrates a significant improvement in Macro F1-score, reaching 0.89, over various CNNs while using 99.8% fewer parameters. It fully decodes three degrees of rat motion (dorsiflexion, plantarflexion, and pricking stimulation), showcasing the potential for efficient and accurate hardware integration. This work highlights a path towards the development of next-generation hardware-assisted neural interfaces.**

***Index Terms*—SNN, neural decoding, peripheral nerve interface, PNS recording, neural front end.**

I. Introduction

The peripheral nervous system (PNS) neural interface represents a groundbreaking technological advancement with the potential to restore motor functions for patients suffering from spinal cord injuries, amputations, and other debilitating conditions. The efficacy of these neural interfaces heavily depends on backend processing, which typically dominates both power consumption and the accuracy of motor decoding.

In contrast to central nervous system (CNS) neural interfaces that are located in the brain [1], [2], PNS neural interfaces [3], [4] are more localized to specific limb functions, as illustrated in Fig. 1. In the context of sensorimotor applications, neural signals propagating through the fascicles within the nerve truck include efferent motor commands and afferent signals encoding proprioceptive and tactile information. Accurate and efficient translation of these signals is crucial for clinical applications. However, current artificial intelligence networks [5] face challenges in achieving this. Convolutional neural networks (CNNs), while powerful, often require substantial computational resources and are not optimized for the high-dimensional, time-domain data typical of neural signals [6].

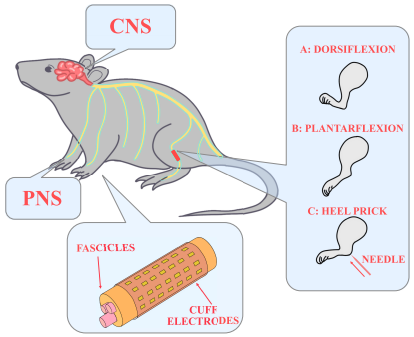

Fig. 1. Illustration of CNS and PNS interfaces, with three degrees of motion.

Spiking neural networks (SNNs) [7], inspired by biological neural networks, offer a promising solution. Unlike traditional neural networks that generate continuous signals, SNNs utilize neurons that produce discrete spikes. This spiking behavior significantly reduces the number of required neurons and enhances efficiency by applying an activation threshold to neural signals, deactivating the network when signals are below this threshold. This makes SNNs particularly well-suited for neural interface applications, as they efficiently process time-domain signals while significantly reducing power consumption [8].

The advantages of SNNs are especially relevant in the context of wearable and implantable devices, where power efficiency and miniaturization are critical. For instance, a high-power-consuming device may require frequent recharging or larger batteries, both of which can limit the practicality and comfort of the device for patients. By reducing power consumption, SNNs can enable more compact, longer-lasting neural interfaces.

This paper proposes a multiplier-less SNN architecture that not only surpasses state-of-the-art accuracy but also fully decodes three degrees of motion observed during experiment with rats. The proposed SNN architecture has been designed to operate with fewer neurons and reduced computational complexity, directly addressing the limitations of traditional approaches. We present a digital implementation of this network, demonstrating its potential to significantly enhance classification efficiency by substantially increasing the Macro F-1 score to 0.89 and reduce the number of parameters by 99.8%, when compared to CNNs, thereby minimizing power consumption in neural interfaces.

Section II introduces the methodology of data processing and network architecture, while Section III illustrates the experimental results and comparisons with conventional CNN models. Section IV presents the concluding remarks. This

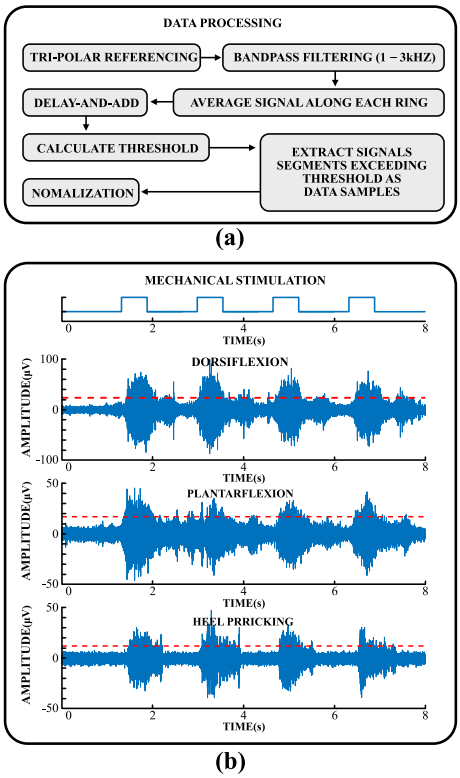

Fig. 2. (a) The sequential data pre-processing steps used in the study, starting with tri-polar referencing and culminating with normalization, (b) The figure illustrates the neural responses to three different stimuli—dorsiflexion, plantarflexion, and heel pricking—following the Delay-and-Add operation, with the threshold levels indicated by red dashed lines.

paper aims to provide valuable insights into the application of hardware-assisted SNNs in sensory activity decoding, offering a path toward more efficient and accurate peripheral nerve interfaces. By leveraging the unique advantages of SNNs, this work highlights a promising direction for developing next-generation neural interfaces that are both high-performing and energy-efficient, ultimately enhancing the quality of life for patients requiring motor function restoration.

II. Materials and Methods

This section outlines the comprehensive methodologies employed in this study, including dataset acquisition, data pre-processing, and the design and implementation of the Spiking Neural Network (SNN) architecture. These steps are essential to ensure accurate and efficient classification of naturally evoked compound action potentials (nCAPs) from peripheral nerve recordings.

A. Dataset Acquisition

This paper leveraged a high-resolution dataset derived from peripheral nerve recordings [9]. The experimental setup involved a multi-channel nerve cuff electrode positioned on the sciatic nerve of nine Long-Evans rats. Each recording session was dedicated to capturing the neural responses elicited by distinct mechanical stimuli—specifically dorsiflexion, plantarflexion, and heel pricking—conducted 100 trials each [9]. These stimuli induced different types of afferent nerve responses, which were recorded and classified in the study, allowing for a detailed analysis of peripheral nerve activity under varied physiological conditions.

B. Data Pre-Processing

To effectively train a SNN, each data sample must capture the full context of naturally evoked compound action potentials (nCAPs) within a uniform time frame. Detailed data pre-processing [9], illustrated in Fig. 2(a), was employed as follows:

1) Tri-polar Referencing: Noise reduction was achieved by subtracting the averages of the first and last electrode rings from each channel.

2) Bandpass Filtering: A 6th-order Butterworth filter with a 1 kHz to 3 kHz passband isolated important frequency components [9], [10].

3) Signal Averaging: Signals from the eight contacts per ring were averaged, reducing the dataset from 56 to 7 channels, each representing one ring.

4) Delay-and-Add: This technique improved the signal-to-noise ratio (SNR) and highlighted nCAPs by delaying the averaged signal from each ring by one timestamp and combining them.

5) Threshold Calculation: Significant spikes were identified using a threshold from [11], set to four times the Median Absolute Deviation (MAD) divided by 0.6745, as shown in Fig. 2(b):

$$\text{Threshold} = \frac{4 \times \text{MAD}}{0.6745} \tag{1}$$

The combined signal was scanned for points where the amplitude exceeded a predefined threshold, identifying potential nCAPs. Each candidate point's corresponding data in the tripole referenced and filtered multichannel recording with its surrounding 99 data points, forming a complete data sample with a uniform time duration, was evaluated and saved in a 56×100 structure.

6) Dataset Construction: Due to physiological variations among rats, datasets were customized for each subject. For example, Rat No. 10 features 5463 nCAPs for dorsiflexion, 6607 for plantarflexion, and 11745 for pricking. To enhance the performance of SNN model, the datasets were balanced using the smallest class as a reference and then split into a training set and a test set at a $7:3$ ratio.

C. Spiking Neural Network Architecture

SNNs are a form of artificial neural networks that aim to mimic the operation of biological neural networks more closely than traditional artificial neural networks (ANNs) [12]. In SNNs, neurons communicate with discrete spikes (or action potentials) which are binary events that occur at specific points in time, rather than continuous values.

The proposed SNN network is a 4-layer structure, as illustrated in Fig. 3(a). The input layer contains only inputs with no neurons, the two hidden layers each has 200 neurons, and the output layer has 3 neurons each corresponding to 3 classes: dorsiflexion, plantarflexion, and heel pricking. Every two consecutive layers are connected by a linear transformation layer with a set of learned weights and biases.

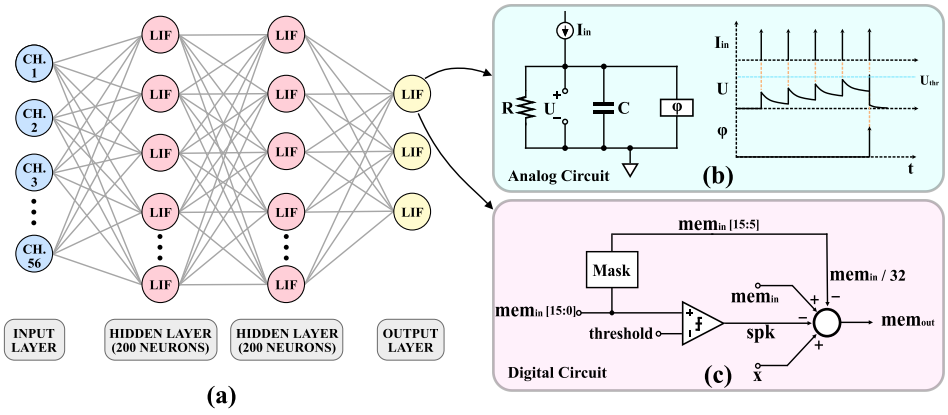

Fig. 3. (a) Spiking Neural Network Architecture, (b) Analog Circuit implementation of Leaky Integrate-and-Fire neuron, when the accumulated input I_{in} over time makes membrane potential U exceeding a certain threshold U_{thr}, the neuron would emit a spike of binary value as output ϕ. (c) Digital implementation of Leaky Integrate-and-Fire neuron

1) LIF Neuron Model: All neurons in the network are modeled as Leaky Integrate-and-Fire (LIF) neurons [13], receiving inputs that simulate an RC (resistor-capacitor) circuit's dynamics [14], as illustrated in Fig. 3(b). These neurons accumulate input over time and emit a binary spike when it exceeds a specified threshold, resetting the input to a lower value [15]. This model simplifies the electrical characteristics of biological neurons.

The LIF model is conceptually similar to an RC circuit with a time-varying input current $I_{\text{in}}(t)$. However, traditional LIF models involve numerous hyperparameters like resistance, capacitance, time step (Δt), and threshold U_{th}), making them complex for deep learning. The proposed model combines various variables into a single hyperparameter, decay rate ($\beta = 0.96875 = 1 - 0.5^5$), with a default threshold 1, as shown in Algorithm 1.

Algorithm 1 Simplified LIF Neuron With Reset Mechanism

1: **procedure** SIMPLIFIED LIF($mem_{\text{in}}, x, \beta = 1 - 0.5^5, threshold = 1$)
2: $spk \leftarrow (mem_{\text{in}} > threshold)$ ▷ Check if spike occurs
3: $mem_{\text{out}} \leftarrow \beta \times mem_{\text{in}} + x - spk \times threshold$ ▷ Update membrane potential
4: **return** spk, mem_{out}
5: **end procedure**

2) Digital Circuit Implementation of LIF Neuron: A simplified LIF model with a reset mechanism and reduced parameters is given in Algorithm 1. Each LIF neuron has four inputs (mem_{in}, x, β, $threshold$) and two outputs (spk, mem_{out}). The voltage potential U across the Capacitor C in Fig. 3(b) is represented by mem_{in} and mem_{out}, which correspond to the current and next state of the LIF neuron respectively. The analog input current I_{in} is represented as x in the digital implementation, which is the weighted sum of neuron outputs from the previous layer. The decay rate β specifies how much mem_{in} will decay to mem_{out} for each step.

At each step the current state of the LIF neuron (mem_{in}) is firstly compared to $threshold$, which is set to 1 in this architecture. If mem_{in} exceeds $threshold$, spk is set to 1; otherwise, it is set to 0. The digital circuit design described in Fig. 3(c) is implemented using a comparator with two inputs (mem_{in} and $threshold$) and one output (spk). The output spk will be weighted and inputted to the neurons in the next layer.

The state of the LIF neuron is subsequently updated using Equation 2. The state mem_{in} is decayed by β and added by input x. If a spike is detected in the cycle, a value of threshold is deducted from the state. The updated state mem_{out} is then used as the input (mem_{in}) for the same neuron in the next cycle.

$$mem_{\text{out}} = \beta \times mem_{\text{in}} + x - spk \times threshold. \quad (2)$$

In the digital circuit implementation, β, spk, and $threshold$ are specifically chosen so that the circuit design is simplified without using multipliers. Specifically, $threshold$ is set to 1 so that Equation 2 can be written as:

$$mem_{\text{out}} = \beta \times mem_{\text{in}} + x - spk. \quad (3)$$

To transform the neuron model to hardware implementation, β is chosen to be $1 - 0.5^5 = 1 - \frac{1}{32}$, which further simplifies the state equation:

$$mem_{\text{out}} = (mem_{\text{in}} - \frac{mem_{\text{in}}}{32}) + x - spk. \quad (4)$$

Equation 4 is the final digital circuit design described in Fig. 3(c), where integer adder and comparator enable unsigned fixed point operations. Furthermore, $mem_{\text{in}}/32$ can be easily achieved by masking out the last 5 bits of the input. With this approach, the digital circuit design of each LIF neuron can be built with only one integer comparator and one integer adder. This design is free of any multipliers, which greatly reduces computation resources and area [18].

D. Training

1) Input Preparation: The SNN employed in this study is designed to process input data x in a shape of $512 \times 100 \times 56$, indicating it processes sequences of 56-channel data over 100 time steps, which is consistent with the sample extracted in pre-processing. The batch size of 512 allows the model to be trained on subsets of the data, reducing the memory requirements and resulting in more frequent updates, which leads to faster learning [19].

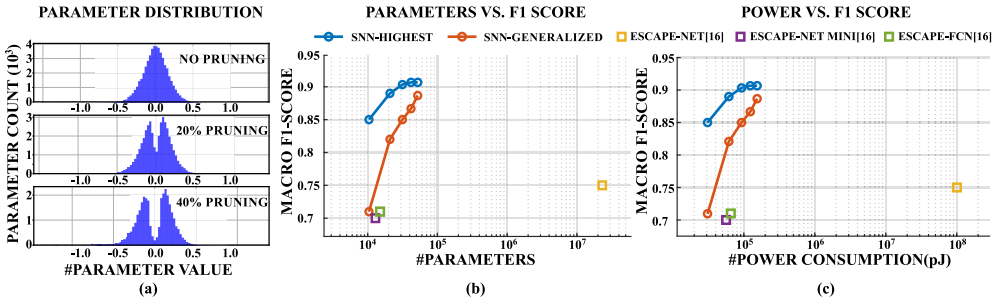

Fig. 4. (a) Distribution plots of parameter values for the SNNs at different levels of pruning: no pruning, 20% pruning, and 40% pruning. As the pruning increases, the distribution of parameter values around zero is decreasing, indicating that less significant weights have being removed. (b) Plot of number of parameters against the Macro F1-scores of SNNs and the CNNs[16], [17]. Note: The data for CNNs is provided for reference only, as the CNNs employed different data processing methods, making direct comparison not possible. (c) This Plot shows the power consumption against the Macro F1-score for the SNNs and the CNNs[16], [17].

TABLE I
CLASSIFICATION PERFORMANCE OF SNN

Class	Precision	Recall	F1-Score	Test set Size
Highest Performance (Rat 9)				
Dorsiflexion	0.95	0.91	0.93	2223
Plantarflexion	0.87	0.92	0.89	2262
Prick	0.91	0.88	0.90	2236
Across Rat 5 to Rat 10				
Dorsiflexion	0.89	0.91	0.90	11252
Plantarflexion	0.88	0.88	0.88	11361
Prick	0.89	0.87	0.88	11118

2) Loss Function and Optimizer: The Cross-Entropy Loss (CE Loss) was used as the loss function for training the SNN. ADAM optimizer [20] was used to update all learnable parameters and minimize the loss function during the training process. The learning rate was $lr = 7.2e-4$, and the exponential decay rates for the moment estimates were $\beta = (0.9, 0.999)$.

III. EXPERIMENT RESULTS

The SNN model for each rat was trained using the same method. The testing results of the model with the highest macro F1-score are given in Table.I. A part of the nCAPs files from each rat, covering all three types, were selected to create a database that is not limited to a single rat. This database was used to training a more generalized SNN and the results are also shown in Table.I. These SNN models were also trained with varying degrees of connectivity reduction, pruning up to 80% of its weights to assess parameter efficiency and performance under scenarios of decreased connectivity. A sample distribution of the models' parameters, both with and without pruning, is depicted in Fig. 4(a).

The performance (Macro F1-scores) of the SNNs and the various CNNs[16], [17] in terms of parameter number is depicted in Fig. 4(b). The results indicate that the Macro F1-score achieved by the SNN in classifying the three types of events exceeds 0.89, with the highest Macro F1-score for an individual rat reaching over 0.9. This high level of performance underscores the substantial advantages of SNN in handling data samples that incorporate time dimension. Despite a significant reduction of 80% in parameters, the SNN maintained over 80% of its initial performance, underscoring its efficiency even with reduced complexity. Although the CNNs[16], [17] is trained with a different dataset due to factors such as the use of 3-fold cross-validation with augmentation and differences in nCAPs extraction, making direct comparisons challenging, the CNN results still provide valuable reference points since both approaches utilized the same raw data. In the SNN, it is clear that there is higher efficiency per parameter. This efficiency is critical for applications requiring minimal computational resources or rapid processing. Compared to ESCAPE-NET, the number of parameters used is reduced by 99.8%, while still maintaining satisfactory performance.

The comparative analysis of power consumption in Fig.4(c), based on data from [18], shows that the SNN architecture is designed for better power efficiency. Under ideal conditions, it can achieve several hundred times lower energy consumption compared to the ESCAPE-NET without compromising performance (Macro F1-score). This efficiency is primarily because the LIF neurons in the SNN output only binary values, leading to multiplications in the fully connected layer that do not consume significant power. The only exception is the multiplication between the neuron's decay constant β and its membrane potential (mem), where $\beta = 0.96875 = 1 - 0.5^5$, which can be achieved using simple shifts and subtraction. This approach significantly reduces the computational complexity and energy consumption of the network.

IV. CONCLUSION

This study presents an approach to enhancing the efficiency and accuracy of peripheral nerve interfaces by leveraging SNNs. The proposed SNN architecture, inspired by biological neural networks, significantly reduces power consumption and computational complexity while maintaining high accuracy in decoding neural signals. The digital implementation of the LIF neuron model further simplifies the design, making it suitable for hardware implementation. Our results highlight the potential of SNNs in fully decoding three degrees of motion observed during experiment with rats, showcasing a significant improvement in Macro F1-score, reaching 0.89, over conventional CNNs while using 99.8% fewer parameters. This efficiency makes SNNs particularly well-suited for applications in wearable and implantable neural interfaces, where power consumption and miniaturization are critical factors.

References

[1] G. O'Leary, J. Xu, L. Long, J. S. Filho, C. Tejeiro, M. ElAnsary, C. Tang, H. Moradi, P. Shah, T. A. Valiante, and R. Genov, "26.2 a neuromorphic multiplier-less bit-serial weight-memory-optimized 1024-tree brain-state classifier and neuromodulation soc with an 8-channel noise-shaping sar adc array," *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, 2020, pp. 402–404. DOI: 10.1109/ISSCC19947.2020.9062962.

[2] F. Shahrokhi, K. Abdelhalim, D. Serletis, P. L. Carlen, and R. Genov, "The 128-channel fully differential digital integrated neural recording and stimulation interface," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 4, no. 3, pp. 149–161, 2010. DOI: 10.1109/TBCAS.2010.2041350.

[3] J. Xu, J. De Sales Filho, E. Hwang, S. Nag, L. Long, M. Kanchwala, M. Abdolrazzaghi, Y. Huang, J. Zariffa, and R. Genov, "Artificially-intelligent fascicle-selective bidirectional peripheral nerve interfaces," *2023 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2023, pp. 1–5. DOI: 10.1109/BioCAS58349.2023.10388693.

[4] J. Xu, J. S. Filho, S. Nag, L. Long, E. Hwang, C. Tejeiro, G. O'Leary, Y. Huang, M. Kanchwala, M. Abdolrazzaghi, C. Tang, P. Liu, Y. Sui, H. You, X. Liu, J. Zariffa, and R. Genov, "Fascicle-selective ultrasound-powered bidirectional wireless peripheral nerve interface ic," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 17, no. 6, pp. 1237–1256, 2023. DOI: 10.1109/TBCAS.2023.3332258.

[5] J. Xu, J. S. Filho, S. Nag, L. Long, C. Tejeiro, E. Hwang, G. O'Leary, Y. Huang, M. Kanchwala, M. Abdolrazzaghi, C. Tang, P. Liu, Y. Sui, X. Liu, G. Eleftheriades, J. Zariffa, and R. Genov, "Fascicle-selective bidirectional peripheral nerve interface ic with 173db fom noise-shaping sar adcs and 1.38pj/b frequency-multiplying current-ripple radio transmitter," *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, 2023, pp. 31–33. DOI: 10.1109/ISSCC42615.2023.10067626.

[6] Z. Cai, H. R. Kalatehbali, B. Walters, M. R. Azghadi, A. Amirsoleimani, and R. Genov, "Stdg: Fast and lightweight snn training technique using spike temporal locality," *2023 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2023, pp. 1–5. DOI: 10.1109/BioCAS58349.2023.10388882.

[7] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997, ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(97)00011-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608097000117.

[8] Z. Cai, H. R. Kalatehbali, B. Walters, M. Rahimi Azghadi, A. Amirsoleimani, and R. Genov, "Spike timing dependent gradient for direct training of fast and efficient binarized spiking neural networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 13, no. 4, pp. 1083–1093, 2023. DOI: 10.1109/JETCAS.2023.3328926.

[9] R. G. Koh, A. I. Nachman, and J. Zariffa, "Classification of naturally evoked compound action potentials in peripheral nerve spatiotemporal recordings," *Scientific Reports*, vol. 9, no. 1, Jul. 2019. DOI: 10.1038/s41598-019-47450-8.

[10] R. Genov, M. Stanacevic, M. Naware, G. Cauwenberghs, and N. Thakor, "16-channel integrated potentiostat for distributed neurochemical sensing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 11, pp. 2371–2376, 2006. DOI: 10.1109/TCSI.2006.884425.

[11] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, "Unsupervised Spike Detection and Sorting with Wavelets and Superparamagnetic Clustering," *Neural Computation*, vol. 16, no. 8, pp. 1661–1687, Aug. 2004, ISSN: 0899-7667. DOI: 10.1162/089976604774201631. eprint: https://direct.mit.edu/neco/article-pdf/16/8/1661/816020/089976604774201631.pdf. [Online]. Available: https://doi.org/10.1162/089976604774201631.

[12] J. K. Eshraghian, M. Ward, E. O. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, "Training spiking neural networks using lessons from deep learning," *Proceedings of the IEEE*, vol. 111, no. 9, pp. 1016–1054, 2023. DOI: 10.1109/JPROC.2023.3308088.

[13] L. F. Abbott, "Lapicque's introduction of the integrate-and-fire model neuron (1907)," en, *Brain Res. Bull.*, vol. 50, no. 5-6, pp. 303–304, Nov. 1999.

[14] S. Dutta, V. Kumar, A. Shukla, N. R. Mohapatra, and U. Ganguly, "Leaky integrate and fire neuron by charge-discharge dynamics in floating-body mosfet," *Scientific Reports*, vol. 7, no. 1, p. 8257, Aug. 2017, ISSN: 2045-2322. DOI: 10.1038/s41598-017-07418-y. [Online]. Available: https://doi.org/10.1038/s41598-017-07418-y.

[15] E. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, 2003. DOI: 10.1109/TNN.2003.820440.

[16] Y.-C. E. Hwang, R. Genov, and J. Zariffa, "Resource-efficient neural network architectures for classifying nerve cuff recordings on implantable devices," *IEEE Transactions on Biomedical Engineering*, vol. 71, no. 2, pp. 631–639, 2024. DOI: 10.1109/TBME.2023.3312361.

[17] R. G. Koh, M. Balas, A. I. Nachman, and J. Zariffa, "Selective peripheral nerve recordings from nerve cuff electrodes using convolutional neural networks," *Journal of Neural Engineering*, vol. 17, no. 1, p. 016042, Jan. 2020. DOI: 10.1088/1741-2552/ab4ac4.

[18] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, *Eie: Efficient inference engine on compressed deep neural network*, 2016. arXiv: 1602.01528 [cs.CV].

[19] M. Li, T. Zhang, Y. Chen, and A. J. Smola, "Efficient mini-batch training for stochastic optimization," *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14, New York, New York, USA: Association for Computing Machinery, 2014, pp. 661–670, ISBN: 9781450329569. DOI: 10.1145/2623330.2623612. [Online]. Available: https://doi.org/10.1145/2623330.2623612.

[20] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].