Mongolian Journal of Engineering and Applied Sciences, 2024, 6(1)

Information and Communication Technology

The Monnum Spiking Dataset for Spike Neural Networks

Adiyabat Enkhjargal[1,*], Byambajav Dorj[1], Bayarpurev Mongol[1], Telmuun Tumnee[1] and Sumiyakhand Dagdanpurev[1] Sandagsuren Dashzeveg[2]

[1]Deparment of Electronics and Communication Engineering, School of Information Technology and Electronics, National University of Mongolia [2]Financial Regulatory Commission of Mongolia

Received on 2024-05-12; Revised on 2024-05-12; Accepted on 2024-05-24

*Corresponding author: Sumiyakhand.D, sumiyakhand@num.edu.mn

Abstract

Spiking Neural Networks are a type of artificial neural network that mimics the way biological neural networks in the brain process information. Spiking neural networks form the foundation of the brain's efficient information processing. While we don't fully understand how these networks calculate, recent optimization techniques allow us to create increasingly complex functional spiking neural networks in a simulated environment. These methods promise to develop more efficient computing hardware and explore new possibilities in understanding brain circuit function. It is essential to have objective methods to compare their performance to speed up the development of such techniques. However, there are currently no widely accepted means of comparing the computational performance of spiking neural networks. We have introduced a new spike-based classification dataset that can be widely used to evaluate software performance and neuromorphic hardware implementations of spiking neural networks to address this issue. To achieve this, we have created a general procedure for converting audio signals into spiking neural network activity, drawing inspiration from neurophysiology. We created the Monnum digit dataset specifically for this study. Within the range of this research, We implemented a digit recognition system from 1 to 10 spoken in the Mongolian language for the Spike neural network. The last is data for training and testing, which was prepared in HDF5 format extension and then trained in the SNN network.

Key words: Spiking neural networks , audio, data set, spoken digits, classification.

1 Introduction

Spiking neural networks (SNNs) are biology's solution for fast and versatile information processing. From a computational point view, SNNs has several desirable properties: They are able to process information in parallel, have a high tolerance for noise and are extremely energy efficient. [1] The precise computations carried out in a given biological SNN largely depend on its connectivity structure. To simulate functional connectivity in silico, a gaining number of training algorithms for SNNs have been developed [2]- [8] for conventional computers and neuromorphic hardware [8]- [14]. The existence of various learning algorithms highlights the need for a systematic way to compare them. Unfortunately, only a few benchmark data sets are widely accepted for SNNs, making it difficult to make accurate comparisons. Hence, in this study, we seek to fill this gap by introducing one new broadly applicable classification data set for SNNs.

2 Methods

To enhance the quantitative comparison among SNNs, we generated a small spike-based classification data set from audio recordings. To achieve our goal, we have collected the MonDigits data set. In the following, we describe the data sets, the audio-to-spike conversion (Section 2.2) and data format used for publication (Section 2.3). We close with a deciption of the SNN model (Section 2.4). The non spikng classifier are explained in Section 2.

2.1 Audio Data Set

In the following, we consider the MD. The MD were designed to prioritize recording quality and precise audio alignment.

1) MonNum Digits: The MD data set consists approximately high-quality recording of spoken digits ranging from zero to ten in Mongolian. In total, 120 speakers were included, fifty three of which were female and sixty seven male. The speaker ages ranged from 10 yr to 40 yr with a mean of 30 yr. We recorded a total 1120 digits in 10—20 sequences for language. (see Fig. 1) The digits were obtained in sequences of ten successive digits. Recordings were performed with a wireless microphone (Sony ECMA4) in a no-shielded room at the National University of Mongolia. Recordings were made in WAVE format with a sample rate of 8kHz and 16-bit precision.

To improve the yield of the automated processing, the raw data audio tracks were manually pre-selected and cut, and then converted to free lossless audio codec format (FLAT) format. The cleaned-up tracks were externally mastered. The cutting times of digit sequences were determined using a gate with speaker-dependent threshold and release time which were optimized by Audacity in [23].

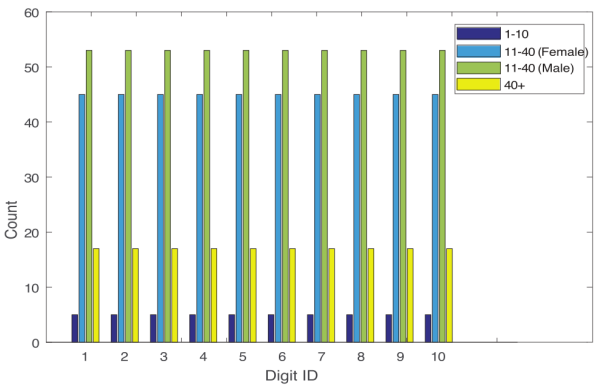

Figure 1: MonNum Digits MD

2.2 Spike Conversion

We used the audio files mentioned earlier to create out spiking data set. Using an artificial model[1] of the inner ear and parts of the ascending auditory pathway, audio data was converted into spikes. (See Fig 2.) This biologically inspired model effectively performs similar signal processing steps as customary spoken language processing applications [16]. First, a hydrodynamic basilar membrane model results spatial frequency dispersion, which is similar to computing a spectrogram with Mel-spaced filter banks. Second, Instantaneous firing rates are computed from separated frequencies using a biologically motivated transmitter pools based hair cell (HC) model. This cell model includes refractory effect and a layer of bushy cells (BCs) that enhance phase locking. (refer to Fig.2) All model parameters were chosen to mimic biological findings, thereby reducing the amount of free parameters [17].

The inner-ear model approximates auditory spiking activity with a low computational cost. This biologically inspired conversion eliminates user-specific audio-to-spike transformation, ensuring comparability and serving as the basis for our benchmark data sets. .

2.3 Event-Based Data format

We simplified access to the data set by using an event-based representation of spikes in the Hierarchical Data Format (HDF5) to making it easier for a broader community to use. This choice ensured short download times and easy access from most common programming environments [17]. For data set, we provide a single HDF5 file which holds spikes, digit labels, and additional meta information. We made these files publicly available[2] , together with supplementary information on the general usage as well as code snippets. A single file is organized as follows [17]

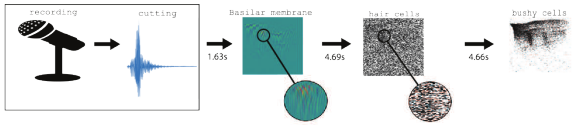

Figure 2: Processing tube for MD

2.4 Spiking Network model

We trained networks of leaky integrate-and-fire neurons using surrogate gradients and backpropagation through time (BPTT) with supervised loss functions to established a performance reference on a spiking data set. The study describes the network architectures (Section 2.4.1), followed by the applied neurons and synapse model (Section 2.4.2). We conclude with a depiction of the weight initialization (Section 2.4.3), the supervised learning algorithm (Section 2.4.4), the loss function (Section 2.4.5), and the regularization techniques (Section 2.4.6).

2.4.1 Network model

The spiketrains emitted by the $N_{ch} = 70$ BCs were used to stimulate the actual classification network [17]. In this article, we trained both feed-forward and recurrent networks; each hidden layer contains $N = 128$ LIF neurons. For all network architectures, the last layer was accompanied by a linear readout consisting of leaky integrators which did not spike.

2.4.2 Neuron and synapse models

Neuron and Synapse Models: We considered LIF neurons where membrane potential $u_i^{(l)}$ of the i th neuron in layer l obeys the differential equation

$$\tau_{mem} \frac{du_i^{(l)}}{dt} = -[u_i^{(l)}(t) - u_{leak}] + RI_i^{(l)}(t) \qquad (1)$$

with the membrane time constant τ_{mem} the input resistance R, the leak potential u_{leak} and the input current $I_i^{(l)}(t)$. Spikes were described by their firing time. The kth firing time of neuron i in layer l is denoted by ${}_k t_i^{(l)}$ and defined by a threshold criterion.

[1] https://github.com/electronicvisions/lauscher

[2] https://www.tensorflow.org

$$ {}_kt_i^{(l)} : u_i^{(l)}({}_kt_i^{(l)}) \geq u_{thres} \tag{2}$$

Immediately after ${}_kt_i^{(l)}$, the membrane potential is set to the leak potential $u_i^{(l)}(t) = u_{leak}$. The synaptic input current onto the i^{th} neuron in layer l was generated by the arrival of presynaptic spikes from neuron $j, S_j^{(l)}(t) = \sum_k \delta(t - {}_k t_j^{(l)})$. A common first-order approximation to model the time course of synaptic currents are exponentially decaying currents which sum linearly [18].

$$\frac{du_i^{(l)}}{dt} = -\frac{I_i^{(l)}(t)}{\tau_{syn}} + \sum_j W_{ij}^{(l)} S_j^{(l-1)}(t) + \sum_j V_{ij}^{(l)} S_j^{(l)}(t) \tag{3}$$

where the sum runs over all presynaptic partners j and $W_{ij}^{(l)}$ are the corresponding afferent weights from layer below. The $V_{ij}^{(l)}$ resemble the recurrent connections within each layer. In this work, the reset was incorporated in (1) through an extra term

$$\frac{du_i^{(l)}}{dt} = -\frac{u_i^{(l)} + u_{leak} - RI_i^{(l)}}{\tau_{mem}} + S_i^{(l)}(t)(u_{leak} - u_t) \tag{4}$$

To formulate the above equation in discrete time for time step n and stepsize δt over a duration $T = n\delta t$, the output spiketrain $S_i^{(l)}[n]$ of neuron i in layer l at time step n is expressed as a nonlinear function of the membrane potential $S_i^{(l)}[n] = \Theta\left(u_i^{(l)} + u_{thres}\right)$ with the Heavyside function Θ. For small time steps δt, we can express the synaptic current in disrete time as follows:

$$I_i^{(l)}[n+1] = kI_i^{(l)}[n] + \sum_j W_{ij}^{(l)} S_j^{(l)}[n] + \sum_j V_{ij}^{(l)} S_j^{(l)}[n] \tag{5}$$

Furthermore, by asserting $u_{leak} = 0$ and $u_{thres} = 0$, the membrane potential can be written compactly as

$$I_i^{(l)}[n+1] = \lambda I_i^{(l)}[n]\left(1 - S_i^{(l)}[n]\right) + (1-\lambda) I_i^{(l)}[n] \tag{6}$$

where we have set $R = (1-\lambda)$ and introduced the constrants $k = \exp\left(-\frac{\delta t}{\tau_{syn}}\right)$ and $k = \exp\left(-\frac{\delta t}{\tau_{mem}}\right)$

2.4.3 Weight Initialization

In all our spiking network simulations we use Kaiming's uniform initialization [19] for the weights W_{ij}, V_{ij}. Specifically, the initial weights were drawn independently from a uniform distribution $\mathcal{U}\left(-\sqrt{k}, \sqrt{k}\right)$ with $k = (\#\text{ afferent connections})^{-1}$.

2.4.4 Supervised Learning

The goal of learning was to minimize a cost function $\mathcal{L}$ over the entire data set. To achieve this, surrogate gradient descent was applied which modifies the network parameters W_{ij}

$$W_{ij} \leftarrow W_{ij} - \eta \frac{\partial \mathcal{L}}{\partial W_{ij}} \tag{7}$$

with the learning rate η. We used custom PyTorch [20] code implementing the SNNs. Surrogate gradients were computed using PyTorch's automatic differentiation capabilities by overloading the derivative of the spiking nonlinearity with a differentiable function as described previously [8], [20]. An instructive example of such an implementation in PyTorch can be found online.[3] Specifically, we chose a fast sigmoid for the surrogate gradient

$$\sigma\left(u_i^{(l)}\right) = \frac{u_i^{(l)}}{1 + \beta\left|u_i^{(l)}\right|} \tag{8}$$

with the steepness parameter β.

2.4.5 Loss functions

We applied a cross entropy loss to the activity of the readout layer $l = L$. On data with N_{batch} samples and N_{glass} classes, $\{(x_s, y_s)|s = 1, \ldots, N_{bartch}; y_s \in \{1, \ldots, N_{glass}\}\}$ it takes the from

$$\mathcal{L} = -\frac{1}{N_{batch}} \sum_{s=1}^{N_{batch}} \mathbb{1}(i = y_s) \log\left(\frac{exp(u_i^{(L)}[\hat{n}_i])}{\sum_{i=1}^{N_{class}} exp(u_i^{(L)}[\hat{n}_i])}\right) \tag{9}$$

with the indicator function $\mathbb{1}$. We tested the following two choices for the time step $\widetilde{n}$: For the max-over-time loss, the time step with maximal membrane potential for each readout unit was considered $\widetilde{n}_i = argmax_n u_i^{(L)}[\widetilde{n}_i]$. In contrast, the last time step T in case of the last-time-step loss. We minimized the cross entropy in (9) using the Adamax optimizer [22].

2.4.6 Regularization

For our experiments, we added synaptic regularization terms to the loss function to avoid pathologically high or low firing rates. In more details. We used two different regularization terms: As a first term, we used a per neuron lower threshold spike count regularization of the form

$$\mathcal{L}_1 = -\frac{S_l}{N_{batch}} \sum_{s=1}^{N_{batch}} \sum_{i=1}^{N} \left[\max\left\{0, \quad \frac{1}{T}\sum_{n=1}^{T} S_i^{(l)}[n] - \theta_l\right\}\right]^2 \tag{10}$$

with strength S_l, and threshold θ_l. Second, we used an upper threshold mean population spike count regularition

$$\mathcal{L}_2 = -\frac{S_u}{N_{batch}} \sum_{i=1}^{N_{batch}} \left[\max\left\{0, \quad \frac{1}{N}\sum_{i=1}^{N}\sum_{n=1}^{T} S_i^{(l)}[n] - \theta_l\right\}\right]^2 \tag{11}$$

3 Results

First, We prepared the spoken digit by cutting it to an average length of 0.85 seconds. We converted our audio WAV files to the FLAC format with a sampling rate of 8KHz using FFmpeg and Audacity open-source software. After that, Audio data were converted into spikes using a model of inner ear and the ascending auditory pathway which combines a basilar membrane (BM) model with population of hair cells (HCs) followed by a population of bushy cells (BCs) for spike generation. These results are shown in the following it. The timing of the audio spike varies from one audio to another. You can see the generation spikes in

Table 1: The process of spike generation

Stage	Spent time
RmsNormalizer	0.0000 seconds.
HanningWindow	0.0012 seconds.
BasilarMembrane	1.7631 seconds.
HairCell	2.9683 seconds.
BushyCell	3.1463 seconds.
Wave2Spike	7.8800 seconds.

the following figure 3. This dataset uses a more sophisticated cochlear model to generate the spike data corresponding to audio recordings of spoken digits.

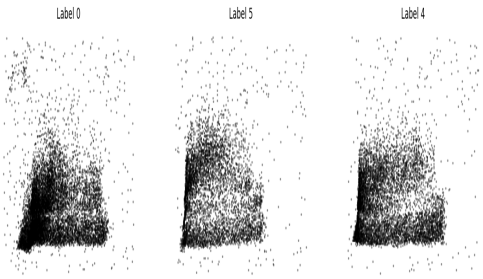

Figure 3: Spikes in 700 input channels were generated using an artificial cochlea model

To determine the relevance of our newly created spiking dataset, we sought to confirm that the dataset was not saturated and to prove that spike timing information was essential for accurate task solving. To conduct the test, we first created a reduced version of the dataset that eliminated all temporal. After reducing spike count data set, we trained various linear and nonlinear support vector machine (SVM) classifiers and measured their classification performance on respective test sets. We observed that the linear SVM overfitted the SMD data, but its test performance marginally exceeded the 54% accuracy (Figure 4). Thus linear classifiers provided a low degree of generalization. We trained SVMs with a radial basis function (RBF) kernel to evaluate whether this was different for the nonlinear classifier. In the following figure, We show the accuracy of the test of the linear SVM machine and nonlinear SVM. A negligibly better performance of about 60% on the SMD was achieved when using a SVM with a radial basis function (RBF) kernel. Next, We considered

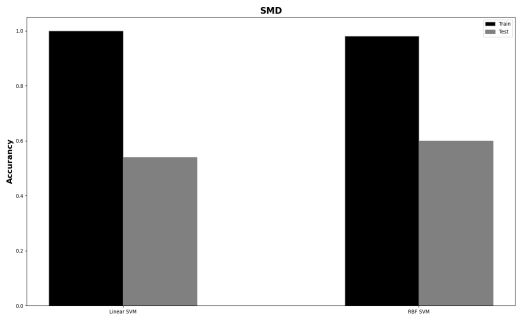

Figure 4: Bar graph of classification accurancy for different SVMs

spiking neural networks (SNNs) with fixed [17], finite time constants on the order of milliseconds inspired by biology. Schematic of a single layer recurrent network with two readout units. We applied two different loss functions for SVM and SNNs: First, a max-over-time loss was considered, where the time step with maximal activity of each readout was used to calculate the cross entropy (marked by colored arrows). Second, a last-time-step loss was utilized where only the last time step of the activation was considered in the calculation of the cross entropy (marked by gray arrow). The inset illustrates the corresponding feed-forward topology. [17].

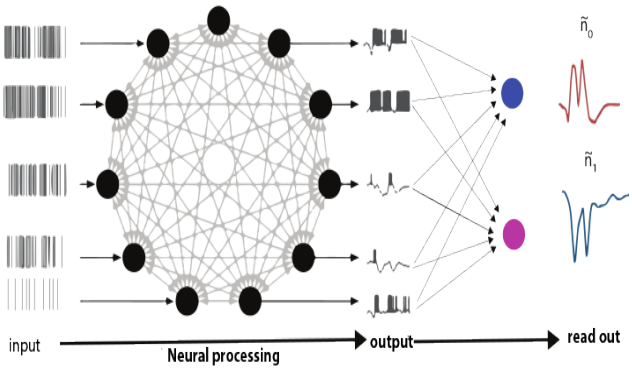

Figure 5: Schematic SNN setup

[3]https://github.com/fzenke/spytorch

We have confirmed that set of spiking data contain valuable temporal information that can be read out by an appropriate classifier. Our goal was to train Spiking Neural Networks (SNNs) using Leaky Integrate-and-Fire (LIF) neurons with Backpropagation Through Time (BPTT) in order to establish initial performance benchmarks and evaluate their ability to generalize. One challenge in training SNNs with gradient descent is the presence of the derivative of the neural activation function in the gradient evaluation. This results in ill-defined gradients due to the discontinuous nature of spiking. To train networks of LIF neurons using supervised loss function, a surrogate gradient approach was employed [8]. Surrogate gradients can be viewed as a continuous relaxation of the natural gradients of a SNN. They can be implemented as an in-place replacement when performing BPTT. Importantly, we did not change the neuron model and the associated forward-pass of the model, but used a fast sigmoid as a surrogate activation function when computing gradients (Methods Section 2.4.4). We trained SNN architecture on the SMD. The SNN achieved the highest accurancy of 64.9% on the SMD. The performance was markedly better than the one reached by SVMs.

4 Conclusions

This paper presents a new dataset of Mongolian-language spoken digits for spike-based classification, facilitating a quantitative comparison of SNNs. Further, we provide a first set of baselines for future comparisons by training spiking and non-spiking classifiers. In order to drive progress in the field of neuromorphic computing, it is essential to establish a set of benchmarks that present real-world challenges. This will allow for the measurement of improvements and the standardization of evaluations across various platforms. We consider the dataset in this article to be our contribution towards this objective. In the future, we can significantly increase the number of datasets and change the parameters of SNN to improve the results. In this summary, we introduced an open spiking dataset of Mongolian-language spoken digits and conducted the first set of performance measurements using the SNN classifier. This marks a significant step towards quantitatively comparing the functionality of SNNs in traditional computers and neuromorphic hardware.

Author Contributions

Adiyabat E. and Dashzeveg S. developed the machine learning method and created the HDF5 dataset. They trained the SSN model. Byambajav D. and Bayarpurev M. provided guidance on the experimental process and the analysis of the results. Sumyakhand D. and Telmuun T. were responsible for writing and editing the article.

Conflict of Interest

There will not be any conflicts of interest.

References

[1] K. Boahen, “A neuromorph’s prospectus,” Comput. Sci. Eng., vol. 19, no. 2, pp. 14 28, 2017.

[2] F. Zenke and S. Ganguli, “SuperSpike: Supervised learning in multilayer spiking neural networks,” Neural Comput., vol. 30, no. 6, pp. 1514–1541, Jun. 2018. DOI: 10.1162/neco_a_01076.

[3] M. Pfeiffer and T. Pfeil, “Deep learning with spiking neurons: Opportunities and challenges,” Frontiers Neurosci., vol. 12, p. 774, Oct. 2018. DOI: .

[4] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, “Deep learning in spiking neural networks,” Neural Netw., vol. 111, pp. 47–63, Mar. 2019. DOI: .

[5] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, “Long short-term memory and learning-to-learn in networks of spiking neurons,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 787–797.

[6] S. B. Shrestha and G. Orchard, “SLAYER: Spike layer error reassign- ment in time,” in Advances in Neural Information Processing Systems, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2018, pp. 1419–1428.

[7] S. Woániak, A. Pantazi, T. Bohnstingl, and E. Eleftheriou, “Deep learning incorporating biologically inspired neural dynamics and in- memory computing,” Nature Mach. Intell., vol. 2, no. 6, pp. 325–336, Jun. 2020. v

[8] E. O. Neftci, H. Mostafa, and F. Zenke, “Surrogate gradient learning in spiking neural networks,” 2019, arXiv:1901.09948. [Online]. Available: http://arxiv.org/abs/1901.09948

[9] J. Schemmel, D. Briiderle, A. Griibl, M. Hock, K. Meier, and S. Millner, “A wafer-scale neuromorphic hardware system for large-scale neural modeling,” in Proc. IEEE Int. Symp. Circuits Syst., May 2010, pp. 1947–1950. DOI: .

[10] S. Friedmann, J. Schemmel, A. Grubl, A. Hartel, M. Hock, and K. Meier, “Demonstrating hybrid learning in a flexible neuromorphic hardware system,” IEEE Trans. Biomed. Circuits Syst., vol. 11, no. 1, pp. 128–142, Feb. 2017. DOI: .

[11] S. B. Furber et al., "Overview of the SpiNNaker system architecture," IEEE Trans. Comput., vol. 62, no. 12, pp. 2454–2467, Dec. 2013. DOI: .

[12] M. Davies et al., "Loihi: A neuromorphic manycore processor with on-chip learning," IEEE Micro, vol. 38, no. 1, pp. 82–99, Jan. 2018.

[13] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neu- romorphic asynchronous processors (DYNAPs)," IEEE Trans. Biomed. Circuits Syst., vol. 12, no. 1, pp. 106–122, Feb. 2018. DOI: .

[14] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," Nature, vol. 575, no. 7784, pp. 607–617, Nov. 2019. [Online]. Available:https://www.nature.com/articles/s41586-019-1677-2

[15] M. Davies, "Benchmarks for progress in neuromorphic computing," Nature Mach. Intell., vol. 1, no. 9, pp. 386–388, Sep. 2019. DOI: .

[16] X. Huang, A. Acero, H.-W. Hon, and R. Reddy, Spoken Language Processing: A Guide to Theory, Algorithm and System Development. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.

[17] B. Cramer, Y. Stradmann, J. Schemmel, "The Hiedelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks", IEEE Transactions on Neural Networks and Learning sytems, VOL.33, NO. 7, JULY 2022. DOI: .

[18] W. Gerstner and W. M. Kistler, Spiking Neuron Models: Single Neurons, Populations, Plasticity. Cambridge, U.K.: Cambridge Univ. Press, 2002. DOI: .

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 1026–1034. DOI: .

[20] A. Paszke et al., "Automatic differentiation in PyTorch," in Proc. NIPS, 2017, p. 5.

[21] F. Zenke and T. P. Vogels, "The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural net- works," bioRxiv, p. 2020.06.29.176925, Jun. 2020. [Online]. Available: https://www.biorxiv.org/content/10.1101/2020.06.29.176925v1

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, arXiv:1412.6980. [Online]. Available: http://arxiv. org/abs/1412.6980

[23] "Audacity: Free Audio Editor and Recorder". audacityteam.org. Archived from the original on March 14, 2011. Retrieved January 5, 2012.

[24] Robert Gütig and Haim Sompolinsky. The tempotron: a neuron that learns spike timing-based decisions. Nat Neurosci, 9(3):420–428, March 2006. ISSN 1097-6256. : 10.1038/nn1643.