Decoding Finger Velocity from Cortical Spike Trains with Recurrent Spiking Neural Networks

Tengjun Liu[1,2,3,*], Julia Gygax[3,4,*], Julian Rossbroich[3,4,*],
Yansong Chua[5], Shaomin Zhang[1,2,†], Friedemann Zenke[3,4,†]

[1]*Qiushi Academy for Advanced Studies, Zhejiang University*, Hangzhou, China
[2]*College of Biomedical Engineering and Instrument Science, Zhejiang University*, Hangzhou, China
[3] *Friedrich Miescher Institute for Biomedical Research*, Basel, Switzerland
[4] *Faculty of Science, University of Basel*, Basel, Switzerland
[5] *China Nanhu Academy of Electronics and Information Technology (CNAEIT)*, Jiaxing, China

[*] These authors contributed equally to this work.
[†] Corresponding authors: shaomin@zju.edu.cn, friedemann.zenke@fmi.ch

***Abstract*—**

***Index Terms*—spiking neural network, brain machine interface, cortical spike train decoding, neuromorphic hardware**

I. Introduction

Brain-machine interfaces (BMIs) enable direct communication between biological neural networks and external devices [1]. Significant progress has been made in invasive BMI technology, allowing volitional control of robotic arms [2], [3] and text generation for communication [4]–[7]. Despite rapid improvements in decoding performance, a central challenge persists: the risk of wound infection due to externally mounted pedestals. Fully implanted BMIs offer a potential solution [8]–[10], but introduce new constraints on energy consumption and heat dissipation of the implants. For instance, the American Association of Medical Instrumentation guidelines stipulate that chronically implanted medical devices must not increase tissue temperature by more than 1°C [11]. Consequently, fully implanted BMIs require a computational substrate capable of delivering reliable and accurate decoding performance within strict latency and power limits.

Spiking neural networks (SNNs) offer an attractive solution for fully implanted BMIs [12]. Their spike-based communication minimizes pre-processing requirements for cortical spike trains (CSTs) and makes them suitable for ultra-low power, low-latency neuromorphic hardware [13], [14]. Several studies have explored the decoding abilities of feed-forward SNNs [8]–[10], [15], but these models achieved only moderate decoding performance on standard benchmarks, such as decoding finger velocity from monkeys performing a reaching task [16]. For instance, Liao, Widmer, Wang, *et al.* [10] developed an energy-efficient feed-forward SNN, which achieved only moderate decoding accuracy (coefficient of determination (R^2) value of 0.45 averaged across two CST recording sessions). The community-driven NeuroBench project [8] proposed additional SNN models for CST decoding, termed *SNN* and *SNN_Flat*, achieving average R^2 values of 0.58 and 0.63, respectively. However, the increased decoding accuracy of the *SNN_Flat* model comes at a substantial energy cost. Consequently, existing feed-forward SNN models fail to simultaneously meet decoding performance and energy efficiency requirements.

Recurrent spiking neural networks (RSNNs), which go beyond simple feed-forward architectures, remain largely unexplored for fully implanted BMIs applications. Here, we address this gap by investigating the decoding performance and energy efficiency of RSNNs on finger velocity decoding from CSTs in monkeys [16].

II. Methods

To study the decoding performance of RSNNs, we iteratively developed two distinct model architectures: First, we explored the maximum decoding performance achievable through end-to-end training of RSNNs, which we refer to as "bigRSNN". Second, to balance decoding performance and energy efficiency for fully implanted BMI applications, we developed a lightweight "tinyRSNN", reducing the number of model parameters by several orders of magnitude.

A. Network architectures

Both models consist of a spiking input layer matching the number of electrode channels in the CST recording, followed by a single recurrent layer of standard leaky integrate-and-fire (LIF) neurons, and a readout layer of non-spiking leaky integrator (LI) neurons (Fig. 1A). All LIF and LI units in the network feature synaptic and membrane dynamics with learnable, unit-specific synaptic and membrane time constants [17]. During training, we optimized the membrane potential of the readout units to match the measured monkey finger velocities. The two network models primarily differ in the size of their hidden and readout layers, with tinyRSNN additionally incorporating techniques that encourage activity and connection sparsity (see Section II-D).

To explore the upper limit of decoding capabilities, we designed bigRSNN with 1024 units in the hidden layer. Mimicking ensemble methods that have been proven successful in

machine learning [18], we incorporated a readout layer with five readout heads. Each head consists of two non-spiking LI neurons corresponding to finger velocities along the X and Y axes, respectively. These readout heads were allowed to have different synaptic and membrane dynamics. The final predicted finger velocities were determined by averaging the predictions across all readout heads.

To better accommodate the resource-constrained setting of realistic BMI applications, we designed tinyRSNN with a limited hidden layer size of 64 recurrently connected LIF neurons. In contrast to the multi-head readout strategy of bigRSNN, tinyRSNN employs a simpler readout layer with only two LI units, one each for X and Y directions.

B. Data pre-processing

Both models were trained on multi-unit activity (MUA) from CST recordings obtained over multiple sessions from two macaque monkeys, Indy and Loco [15], [16] (Fig. 1B). For each session, we split the data into 65% training, 10% validation, and 25% test sets. We discretized input spikes into 4 ms long bins, each representing one time step for the model. To enable mini-batch training, we divided the training and validation data into overlapping two-second samples (500 time steps per sample). In contrast, we presented the test data to the models continuously, time step by time step as a single long sample, mimicking real-time processing conditions.

C. Training

The training process for bigRSNN and tinyRSNN followed the same structured approach: We initialized models in the fluctuation-driven regime [19]. For each monkey, we first pre-trained the models for 100 epochs on concatenated and shuffled data samples from all available sessions [16]. After pre-training, we independently fine-tuned the models for 200 epochs on each of three sessions per monkey[1] [15] (cf. Fig. 1D). For each session, we trained five models with different parameter initializations. All SNNs were trained using surrogate gradient descent [20] to minimize the root mean squared error between predicted and target finger velocities. We used the SMORMS3 optimizer [19], [21] with a cosine learning rate schedule to accelerate learning. To avoid silent neurons during training, we added a homeostatic activity regularization term to the loss function [19]. Finally, we applied dropout with $p = 0.3$ to the input and hidden layers to mitigate overfitting. Detailed hyperparameters and training settings for both models can be found in the configuration files in our GitHub repository[2]. All simulations were run in Python version 3.10.12 using custom code based on the Stork SNN simulator [19] and PyTorch [22].

[1]Sessions: I1: indy_20160622_01, I2: indy_20160630_01, I3: indy_20170131_02, L1: loco_20170210_03, L2: loco_20170215_02, and L3: loco_20170301_05.

[2]Code repository: https://github.com/fmi-basel/neural-decoding-RSNN

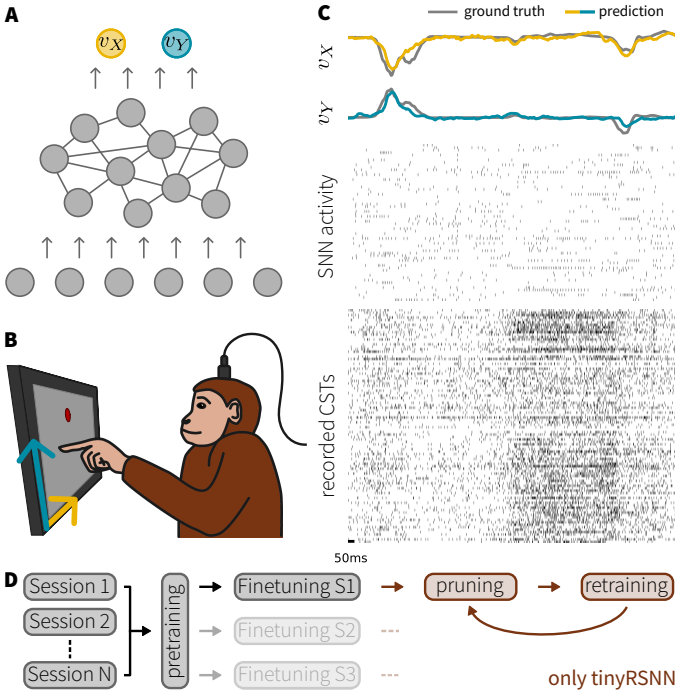

Figure 1. **Setup and example network activity. A)** Schematic of the RSNN network architecture. **B)** The publicly available dataset consists of CSTs recorded from two macaque monkeys performing a self-paced reaching task using chronically implanted electrodes. Recording sites were either in the primary motor cortex (M1) (*Indy*) or in M1 and the primary sensory cortex (S1) (*Loco*) [16]. **C)** Example network activity for tinyRSNN. Bottom: Spike raster of recorded CSTs which serves as input to the model. Middle: Spike raster of the recurrent hidden layer activity. Top: Membrane potentials of the readout units (colored) and the ground-truth finger velocities (gray). **D)** Schematic of the training curriculum. For each monkey, we pre-trained on all available sessions, with subsequent session-wise fine-tuning. For tinyRSNN, we further added iterative pruning at the end of the curriculum.

D. Activity regularization and pruning

We further optimized the tinyRSNN model for energy efficiency through three approaches: First, we encouraged sparse neuronal activity by incorporating an activity regularization term to the loss function, which imposes an upper bound on the average firing rate of the hidden layer [19]. Second, we implemented an iterative pruning strategy for all synaptic weights to reduce the number of parameters and synaptic operations. Following fine-tuning, we initially pruned the 40% of the weights with the smallest magnitude and trained the remaining weights for an additional 100 epochs. We then iteratively pruned an additional 10% of weights and fine-tuned, as long as the R^2 value stayed above a predefined threshold, i.e. 2% below the original accuracy before pruning. Once this threshold was reached, we reduced the pruning rate to 5% and continued until the R^2 value fell below the threshold again (Fig. 1D). Finally, to further reduce the memory footprint of the network, we converted all model parameters and buffers to half precision after training.

III. Results

We assessed the trained models' decoding performance and computational complexity using the following metrics as suggested by the NeuroBench project [8]: For decoding performance, we calculated the R^2 between predicted and observed finger velocities on the test dataset. To evaluate computational cost, we measured the total memory footprint of the networks in bytes, their activation and connection sparsity, and the number of synaptic operations used by the models. The latter was determined using the total number of operations during inference (Dense) and the total number of effective accumulate (AC) and multiply-and-accumulate (MAC) operations. Notably, as both models are pure SNN models without normalization layers, they perform no effective MAC operations. If not stated otherwise, we report averages across five different random initializations $\pm$ standard deviation.

A. Decoding performance

We first assessed the decoding performance of tinyRSNN and bigRSNN and found that both models provided good estimates of finger velocity (Fig. 2). In particular, both models outperformed previously published feed-forward SNN decoders (see Table I), as well as conventional artificial neural networks (ANNs) trained on the same dataset (average R^2 values: *ANN*: 0.579, *ANN_Flat*: 0.615 [8]). As anticipated, the less constrained bigRSNN exhibited superior performance compared to tinyRSNN across all sessions (Table I, Fig. 2B). These results suggest that the incorporation of recurrent connections enhances the processing capabilities of SNNs, conferring advantages in CST decoding tasks.

Table I
Session Specific R^2 Scores Compared to Baseline Models.

Session	Baseline [8]		Ours	
	SNN	**SNN_Flat**	**tinyRSNN**	**bigRSNN**
I1	0.677	0.697	0.752 ± 0.003	0.770 ± 0.003
I2	0.501	0.577	0.545 ± 0.004	0.585 ± 0.012
I3	0.599	0.652	0.746 ± 0.007	0.772 ± 0.006
L1	0.571	0.623	0.622 ± 0.003	0.698 ± 0.006
L2	0.515	0.568	0.608 ± 0.006	0.629 ± 0.008
L3	0.620	0.681	0.690 ± 0.006	0.734 ± 0.005
Mean	**0.581**	**0.633**	$\mathbf{0.660 \pm 0.002}$	$\mathbf{0.698 \pm 0.002}$

B. Computational complexity of the tinyRSNN model

Next, we investigated the memory footprint and computational cost of the models. We found that tinyRSNN had a smaller memory footprint and higher connection sparsity compared to feed-forward resource-optimized SNN decoders [8], despite its superior decoding performance (cf. Table II). The tinyRSNN model uses fewer ACs on average, a direct consequence of activity regularization and pruning. More precisely, it featured an average connection sparsity of 45% and an average activation sparsity of 98.36% (Table II; cf. Tables III and IV for session-specific results). To estimate energy consumption on optimized hardware, we computed the

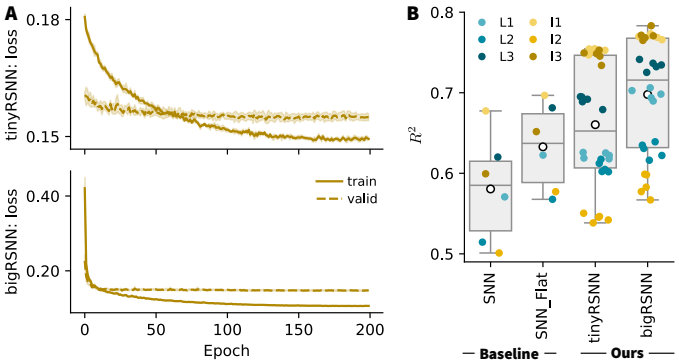

Figure 2. **Learning and decoding performance of the RSNN models. A)** Learning curves for the tinyRSNN (top) and the bigRSNN (bottom) for session I3. **B)** R^2 values for ours and the baseline models (*SNN*, *SNN_Flat* [8]). Each point corresponds to a network with a different random initialization; colors indicate the different sessions.

Table II
Computational Cost of tinyRSNN Compared to Resource-Efficient Baseline Models. Bold: Best Values.

		Baseline [8]		Ours
		SNN	**ANN**	**tinyRSNN**
Memory footprint [bytes]		29248	27160	$\mathbf{27144 \pm 0}$
Connection Sparsity		0	0	$\mathbf{0.45 \pm 0.01}$
Activation Sparsity		**0.9976**	0.6755	0.9836 ± 0.0001
SynOps	**Dense**	7300	**6237**	13440 ± 0
	Effective MACs	**0**	4967	$\mathbf{0 \pm 0}$
	Effective ACs	414	**0**	304 ± 8
Effective Energy Ratio		1.36	506.50	**1**

effective energy ratio [23], [24]. By this metric, tinyRSNN reduced energy consumption by 26.6% compared to the SNN decoder from Yik, Van den Berghe, den Blanken, *et al.* [8]. Furthermore, it requires approximately 500 times less energy than a traditional ANN decoder trained on the same task [8] (see Table II). These results suggest that small RSNN models can effectively leverage activity regularization and synaptic pruning strategies to substantially reduce computational costs while maintaining high decoding performance.

IV. Discussion & Conclusion

We developed two RSNN models for decoding finger velocities from CSTs and evaluated their decoding performance and energy efficiency. Our bigRSNN model significantly improved decoding performance compared to existing models. Additionally, we introduced the tinyRSNN model, tailored for fully implanted BMI applications. This model was designed to find a Pareto optimum between decoding accuracy and energy efficiency. Our work demonstrates that, through the application of suitable training techniques, the computational complexity of RSNNs can be remarkably reduced. By combining small layer sizes, activity regularization, and synaptic pruning, we substantially reduced the computational requirements while hardly impacting decoding performance. Notably, tinyRSNN

exhibits considerable performance improvements over existing models with comparable resource requirements.

While we have demonstrated promising performance improvements, several questions remain open. For instance, the specific roles of individual factors, such as recurrent connections, in driving these improvements remain unclear. A detailed analysis of their impact is a key area for future work. Recent work has highlighted synaptic delays as an architectural feature that can enhance SNN performance [25], [26]. These delays may also allow for reduced memory requirements while being amenable to efficient hardware implementations [27], [28]. In future work, we will explore whether incorporating synaptic delays could further improve CSTs decoding. Although tinyRSNN constitutes a parsimonious model that should in principle be suitable for a hardware implementation, we have not formally shown its performance gains on physical hardware. We intend to implement this model on suitable neuromorphic processors in future work, a crucial step toward realizing practical, fully implanted BMIs.

In conclusion, RSNNs have proven to be competitive models for real-time CSTs decoding that outperform existing models. Moreover, we have shown that their computational complexity can be substantially reduced through suitable training strategies, with minimal impact on accuracy. These advancements bring us closer to the next generation of efficient, high-performance neural decoders for implantable BMIs applications, with the potential to elevate the standard of patient care.

Table III
QUANTIFICATION OF MEMORY FOOTPRINT AND SPARSITY.

	Session	tinyRSNN	bigRSNN
Memory footprint [bytes]	I1 - I3	21 000	4 636 752
	L1 - L3	33 288	5 029 968
	Mean	**27 144**	**4 833 360**
Connection Sparsity	I1	0.47 ± 0.02	0
	I2	0.45 ± 0.03	0
	I3	0.50 ± 0.00	0
	L1	0.44 ± 0.04	0
	L2	0.42 ± 0.02	0
	L3	0.45 ± 0.03	0
	Mean	$\mathbf{0.45 \pm 0.01}$	**0**
Activation Sparsity	I1	0.9838 ± 0.0004	0.9622 ± 0.0002
	I2	0.9853 ± 0.0003	0.9718 ± 0.0006
	I3	0.9842 ± 0.0002	0.9721 ± 0.0002
	L1	0.9831 ± 0.0001	0.9677 ± 0.0012
	L2	0.9832 ± 0.0002	0.9674 ± 0.0010
	L3	0.9820 ± 0.0003	0.9686 ± 0.0014
	Mean	$\mathbf{0.9836 \pm 0.0001}$	$\mathbf{0.9683 \pm 0.0005}$

Table IV
SESSION SPECIFIC AND AVERAGE SYNAPTIC OPERATIONS.

	Session	tinyRSNN	bigRSNN
Dense	I1 - I3	10 368	1 157 120
	L1 - L3	16 512	1 255 424
	Mean	**13 440**	**1 206 272**
Effective ACs	I1	299 ± 14	$48\,096 \pm 254$
	I2	197 ± 12	$34\,837 \pm 659$
	I3	143 ± 2	$33\,289 \pm 181$
	Mean Indy	$\mathbf{213 \pm 4}$	$\mathbf{38\,741 \pm 190}$
	L1	354 ± 28	$44\,664 \pm 1258$
	L2	404 ± 20	$45\,945 \pm 1097$
	L3	427 ± 28	$45\,189 \pm 1437$
	Mean Loco	$\mathbf{395 \pm 19}$	$\mathbf{45\,266 \pm 1056}$

V. ACKNOWLEDGMENTS

This work was supported by STI 2030-Major Projects (2022ZD0208604, 2021ZD0200300), ZJU Doctoral Graduate Academic Rising Star Development Program (2023059), EU's Horizon Europe Research and Innovation Programme (Grant Agreement No. 101070374, CONVOLVE) funded through SERI (Ref. 1131-52302), the Swiss National Science Foundation (Grant Number PCEFP3_202981), and the Novartis Research Foundation.

REFERENCES

[1] U. Chaudhary, N. Birbaumer, and A. Ramos-Murguialday, "Brain–computer interfaces for communication and rehabilitation," *Nature Reviews Neurology*, vol. 12, no. 9, pp. 513–525, 2016.

[2] L. R. Hochberg, D. Bacher, B. Jarosiewicz, *et al.*, "Reach and grasp by people with tetraplegia using a neurally controlled robotic arm," *Nature*, vol. 485, no. 7398, pp. 372–375, 2012.

[3] S. N. Flesher, J. E. Downey, J. M. Weiss, *et al.*, "A brain-computer interface that evokes tactile sensations improves robotic arm control," *Science*, vol. 372, no. 6544, pp. 831–836, 2021.

[4] C. Pandarinath, P. Nuyujukian, C. H. Blabe, *et al.*, "High performance communication by people with paralysis using an intracortical brain-computer interface," *elife*, vol. 6, e18554, 2017.

[5] F. R. Willett, D. T. Avansino, L. R. Hochberg, J. M. Henderson, and K. V. Shenoy, "High-performance brain-to-text communication via handwriting," *Nature*, vol. 593, no. 7858, pp. 249–254, 2021.

[6] D. A. Moses, S. L. Metzger, J. R. Liu, *et al.*, "Neuroprosthesis for Decoding Speech in a Paralyzed Person with Anarthria," *New England Journal of Medicine*, vol. 385, no. 3, pp. 217–227, Jul. 2021.

[7] N. S. Card, M. Wairagkar, C. Iacobacci, *et al.*, "An accurate and rapidly calibrating speech neuroprosthesis," en, *The New England journal of medicine*, vol. 391, no. 7, pp. 609–618, Aug. 2024.

[8] J. Yik, K. Van den Berghe, D. den Blanken, *et al.*, "NeuroBench: A Framework for Benchmarking Neuromorphic Computing Algorithms and Systems," *NeuroBench*, vol. 2304.04640, Jan. 2024.

[9] E. A. Taeckens and S. Shah, "A spiking neural network with continuous local learning for robust online brain

machine interface," *Journal of Neural Engineering*, vol. 20, no. 6, p. 066 042, Jan. 2024.

[10] J. Liao, L. Widmer, X. Wang, *et al.*, "An energy-efficient spiking neural network for finger velocity decoding for implantable brain-machine interface," in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, IEEE, 2022, pp. 134–137.

[11] P. D. Wolf and W. Reichert, "Thermal considerations for the design of an implanted cortical brain–machine interface (bmi)," *Indwelling Neural Implants: Strategies for Contending with the In Vivo Environment*, pp. 33–38, 2008.

[12] J. Dethier, P. Nuyujukian, C. Eliasmith, *et al.*, "A Brain-Machine Interface Operating with a Real-Time Spiking Neural Network Control Algorithm," *Advances in Neural Information Processing Systems*, vol. 24, 2011.

[13] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, *et al.*, "Neuromorphic Silicon Neuron Circuits," *Frontiers in Neuroscience*, vol. 5, May 2011.

[14] E. Donati and G. Valle, "Neuromorphic hardware for somatosensory neuroprostheses," en, *Nature Communications*, vol. 15, no. 1, p. 556, Jan. 2024, Number: 1 Publisher: Nature Publishing Group, ISSN: 2041-1723. DOI: 10.1038/s41467-024-44723-3.

[15] B. Zhou, P.-S. V. Sun, J. Yik, *et al.*, *IEEE BioCAS 2024 Grand Challenge on Neural Decoding for Motor Control of non-Human Primates*, May 31, 2024. DOI: 10.21227/bp1f-te92.

[16] J. E. O'Doherty, M. M. B. Cardoso, J. G. Makin, and P. N. Sabes, *Nonhuman Primate Reaching with Multichannel Sensorimotor Cortex Electrophysiology*, May 2017.

[17] N. Perez-Nieves, V. C. H. Leung, P. L. Dragotti, and D. F. M. Goodman, "Neural heterogeneity promotes robust learning," *Nature Communications*, vol. 12, no. 1, p. 5791, Dec. 2021.

[18] L. Rokach, "Ensemble methods for classifiers," in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Boston, MA: Springer US, 2005, pp. 957–980.

[19] J. Rossbroich, J. Gygax, and F. Zenke, "Fluctuation-driven initialization for spiking neural network training," *Neuromorphic Computing and Engineering*, vol. 2, no. 4, p. 044 016, Dec. 2022.

[20] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, Nov. 2019.

[21] S. Funk, *RMSprop loses to SMORMS3 - Beware the Epsilon!* https://sifter.org/simon/journal/20150420.html, 2015. (visited on 04/20/2022).

[22] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035.

[23] B. Yin, F. Corradi, and S. M. Bohté, "Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks," *Nature Machine Intelligence*, vol. 3, no. 10, pp. 905–913, Oct. 2021.

[24] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb. 2014, pp. 10–14.

[25] M. Zhang, J. Wu, A. Belatreche, *et al.*, "Supervised learning in spiking neural networks with synaptic delay-weight plasticity," *Neurocomputing*, vol. 409, pp. 103–118, 2020.

[26] I. Hammouamri, I. Khalfaoui-Hassani, and T. Masquelier, *Learning Delays in Spiking Neural Networks using Dilated Convolutions with Learnable Spacings*, Jun. 2023. arXiv: 2306.17670 [cs].

[27] S. D'Agostino, F. Moro, T. Torchet, *et al.*, "DenRAM: Neuromorphic dendritic architecture with RRAM for efficient temporal processing with delays," *Nature Communications*, vol. 15, no. 1, p. 3446, Apr. 2024.

[28] F. Moro, P. V. Aceituno, L. Kriener, and M. Payvand, *The Role of Temporal Hierarchy in Spiking Neural Networks*, Jul. 2024. arXiv: 2407.18838 [cs].