

Using Spiking Forward Propagation Through Time to Improve Sign-Language Recognition

Bernardo C. Oradovschi, Fernanda D. V. R. Oliveira and José Gabriel R. C. Gomes

Abstract—This paper investigates the use of Spiking Forward Propagation Through Time (SFPTT) for sign-language recognition on the SL-Animals dataset. Sign-language recognition requires accurate modeling of spatial and temporal patterns, making it well suited for Spiking Neural Networks (SNNs), which naturally process sequential information through spike-based dynamics. Compared with conventional neural networks, SNNs offer sparse event-driven computation, lower memory usage, and potential energy efficiency. Experimental results show that SFPTT significantly outperforms previous SNN training methods, reaching 92.76% average validation accuracy and demonstrating strong potential for efficient real-time gesture recognition systems.

Keywords—Spiking Neural Network, Event-driven sensor, Neuromorphic computing

I. INTRODUCTION

Sign-Language recognition is a challenging task, since it requires both spatial and temporal information from sequential data. Traditional deep learning approaches, *e.g.* Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), have achieved state-of-the-art performance, but they require high energy consumption and have high memory requirements, which can be a problem in this large-language-model era [1].

The motivation to use spikes in the sign-language domain has to do with the dependence of hand movement over time, and since spiking neurons use a membrane potential and voltage threshold, they can capture this temporal dependence in a biologically inspired manner. On the subject of memory requirement and energy consumption, since spikes rely on event-driven data, neurons will pass information only when some activity is detected. This results in a data structure that is highly sparse by nature. With proper encoding, sparsity can be a great alternative for decreasing energy consumption and memory requirements.

Why spikes? The fundamental distinction between neuromorphic and conventional vision systems begins at the sensor level. Traditional cameras capture images as a series of fixed-interval frames, sampling the entire visual field regardless of whether a change has occurred. This approach leads to significant data redundancy and high bandwidth requirements,

particularly in tasks like sign-language recognition where background elements often remain static while the hands move rapidly. Conversely, event-based sensors, such as Dynamic Vision Sensors (DVS) [2], operate asynchronously by capturing changes in light intensity at each individual pixel only when they occur. This results in a stream of events that sharply represent temporal and spatial changes, effectively compressing the visual information into a sparse, high-speed data structure that inherently captures the high-frequency dynamics of gestures without the motion blur associated with frame-based sampling.

Processing these asynchronous event streams with conventional Artificial Neural Networks (ANNs) typically requires temporal “binning” into frames, which sacrifices the fine-grained temporal precision inherent to the raw sensor data. Spiking Neural Networks (SNNs) provide a more biologically plausible and efficient computational framework, by processing information through discrete, event-triggered spikes. Unlike the dense, continuous activations in conventional CNNs, SNN neurons continuously update their internal membrane potential, but only transmit information when that potential exceeds a specific voltage threshold [3]. This sparse communication allows for extremely low power consumption and low-latency inference, as computation is driven by activity in the scene rather than by a fixed frame rate. By leveraging the natural synergy between DVS sensors and SNN architectures, recognition systems can achieve more robust spatio-temporal modeling for complex real-time assistive applications [4].

Starting Point: Many deep learning techniques and frameworks can also be applied to SNNs, which means that we can use PyTorch’s extensive functionalities [5]. Some methods to train SNNs, generally classified into shadow training, convert a conventionally-trained neural network into a SNN by changing the activations into spikes [6]. Other methods implement backpropagation using spikes, which raises the problem of non-differentiability of the spike function. Yet other methods use local learning rules for weight updates, which tend to be more biologically plausible, given the unlikelihood of the human brain performing error backpropagation.

This work benchmarks the promising Spiking Forward Propagation Through Time (SFPTT), proposed by [7], which is an adaptation into the spike domain of the Forward Propagation Through Time (FPTT) [8], in a dataset of biomorphic signals from address-event CMOS imagers, to obtain data to support the benefits of using this alternative method for sign-language recognition. Developed code is available at [9].

II. THEORETICAL FOUNDATION

For the literature review, we explore through the foundation of SNNs, like a core spiking neuron, the challenges of non-

Bernardo C. Oradovschi, Fernanda D. V. R. Oliveira and José Gabriel R. C. Gomes are with the Electrical Engineering Program (PEE/COPPE), Federal University of Rio de Janeiro, Rio de Janeiro-RJ, e-mail: [bernardoco,fernanda.dvro,josegabriel]@poli.ufrj.br. This work was supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq/Brazil), grant numbers 309089/2022-0 and 303761/2026-1, by Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ), grant numbers E-26/210.398/2024 and E-26/204.569/2024, by PR-2/UFRJ (ALV 2023), and by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) Finance Code 001.

differentiability of the spike function, and dive more thoroughly on the FPTT and SFPTT training methods and how they bring more performance on sign-language recognition.

A. Spiking Neural Networks

To handle spike signals in the input, we need to make adjustments from the traditional neural networks. A very common spiking neuron is called the Leaky Integrate-and-Fire (LIF) neuron [10].

The name LIF comes from the fact that the neuron has a leaky component that makes the membrane potential decay over time, and it has an integration aspect of accumulating spikes. LIF's functionality can be seen in Figure 1, which depicts the membrane potential reacting to input spike trains, and firing after reaching a certain voltage threshold.

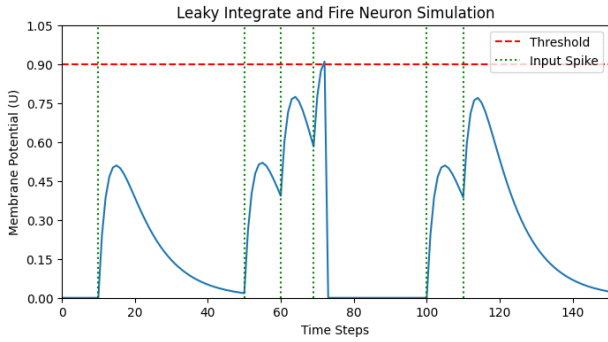


Fig. 1. LIF neuron's membrane potential $U(t)$ receiving stimulus from spike trains. Adaptation from [5].

The membrane potential U is a function that depends on:

- U_{t-1} : The membrane potential at the last time step;
- X_t : Input spike signal;
- θ_t : Weight vector that scales input X_t ;
- $V_{threshold}$: If $U_t > V_{threshold}$, the reset mechanism sets U_t to a resting potential.

B. Spike Function Non-Differentiability

Since spikes are essentially unitary impulses, a single spike can be defined from a delayed Dirac delta function, and since the derivative of the Dirac delta function is the Heaviside step function, there is no derivative at $x = 0$. That causes the function to be not differentiable.

During weight update:

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial S_{out}} \frac{\partial S_{out}}{\partial U} \frac{\partial U}{\partial \theta}, \quad (1)$$

where L is the loss function and S_{out} is the output spike. From [5], we know that $\frac{\partial S_{out}}{\partial U}$ is always 0, except when $U = V_{threshold}$, when it tends to ∞ . This effect implies that there will never be any learning.

To devise a solution for this issue, the use of surrogate gradients, like *arctan* or Difference of Gaussians (DoG), has been proposed for weight updates when backpropagating the error across layers, while still keeping the spike function on the forward pass [11]. This approach seems to be widely used within the academy [5], [7], [12], and [13].

C. Forward Propagation Through Time

The FPTT method, originally proposed by [8], serves as an online learning alternative to backpropagation. We briefly review FPTT in this section, for the sake of completeness. In contrast with Backpropagation Through Time (BPTT), in which the gradient is calculated throughout the entire input sequence length, FPTT updates parameters at each timestep t , that goes from 0 to the sequence length, using instantaneous (x_t, y_t) . It optimizes a *dynamic* loss function l_t^{dyn} that includes two terms: the ordinary objective function L_t and a dynamic regularization penalty term R_t .

$$l_t^{dyn} = L_t(y_t, \hat{y}_t) + R_t(\theta_t), \quad (2)$$

where the objective function L_t is the instantaneous error between the predicted output $\hat{y}_t$ and ground truth y_t , while the regularization function is evaluated and updated as:

$$R_t(\theta_t) = \frac{\alpha}{2} \|\theta_t - \bar{\theta}_t - \frac{1}{2\alpha} \nabla l_{t-1}(\theta_t)\| \quad (3)$$

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} l(\theta)|_{\theta=\theta_t} \quad (4)$$

$$\bar{\theta}_{t+1} = \frac{1}{2} (\bar{\theta}_t + \theta_{t+1}) - \frac{1}{2\alpha} \nabla l_t(\theta_{t+1}) \quad (5)$$

The regularization term uses the running average of previous weights θ_t adjusted by the previous gradient ∇l_{t-1} to impose a smooth parameter update. This is critical for stability during convergence, since it avoids disturbance associated with high instantaneous losses. FPTT updates parameters on every timestep t . The hyperparameter α controls the regularization impact: a low value leads to high impact of the instantaneous loss, possibly resulting in divergence.

FPTT also introduces time constants τ , for U_t updates. This parameter handles the membrane potential leakage, fundamentally allowing the neuron to either keep or forget past information, by controlling how fast the current potential goes to its resting state. On Eq. (4), η is the learning rate that scales the gradient for parameter updates.

Since we need instantaneous (x_t, y_t) in order to calculate the loss at each timestep, the original FPTT introduced approximations for classification tasks with terminal predictions, which means that samples have a single label y at the end of the sequence. To try to reduce the distance between the prediction and labeled distribution, intermediate losses L_t were introduced in [8]:

$$L_t = \frac{t}{T} L_t^{CE} + (1 - \frac{t}{T}) L_t^{KL} \quad (6)$$

$$L_t^{CE} = - \sum_{\bar{y} \in Y} 1_{\bar{y}=y} \log \hat{P}(\bar{y}) \quad (7)$$

$$L_t^{KL} = - \sum_{\bar{y} \in Y} Q(\bar{y}) \log \hat{P}(\bar{y}), \quad (8)$$

where t is the timestep and T is the sequence length, ranging from 10 to 100. This intermediate loss is a combination of two terms: the cross-entropy loss L_t^{CE} and a Kullback-Leibler divergence L_t^{KL} , where $\hat{P}(x)$ is the current estimate of the label distribution and $Q(x)$ is the estimate at the last training

epoch. This means that, at the start of the hand signal, when we do not have much information to perform the hand-signal classification, the divergence term has a bigger impact on the intermediate loss. And, as $t \rightarrow T$, the cross-entropy brings more contribution to the loss.

D. Spiking Forward Propagation Through Time

In order to adapt FPTT principles to the spiking domain, to handle complex spatio-temporal tasks like sign-language recognition, SFPTT requires a few adjustments.

Similar to what happens with RNNs, a straightforward application of FPTT to SNNs fails, and it was found that it requires a forget gate structure in order to converge into a local minima [7].

In terms of bringing the dynamics of Liquid Time Constant (LTC) [14], which uses previous hidden states during weight updates, similarly to LSTMs forget gate functionality. The network then has learnable parameters, which makes it dynamic at the input and at network state, and thus adaptable for longer sequences.

Since LTC is a RNN-specific technique, the authors of [7] proposed the Liquid Spiking Neuron (LSN), which provides adaptations to enable the use of LTC in SFPTT. They also showed that LTC-SNN outperformed BPTT in multiple long sequence datasets.

III. METHODOLOGY

To investigate the robustness of the SFPTT model beyond the standard IBM DVS Gestures benchmark [15], this study utilizes the SL-Animals dataset [16] for evaluation. While high scores on traditional benchmarks are promising, the transition to SL-Animals allows for a critical assessment on how well these results extrapolate to diverse conditions, with 19 distinct hand signals performed by non-fluent users under varying lighting conditions. Because sign-language recognition demands high spatiotemporal awareness, SL-Animals provides this need for challenging model complexity. By utilizing a dataset also benchmarked by [17], this work yields a comparative experiment for evaluating the SFPTT's performance against the performance of existing state-of-the-art techniques.

A. Dataset

A well-known dataset for benchmarking techniques on hand movement data is the DVS Gestures, in which the SFPTT achieved a very high score [7]. To see if these results extrapolate to similar data, in this work, the algorithms are benchmarked on the SL-Animals dataset, which is sign-language specific. This dataset is also used for benchmarking [17], so we can compare SFPTT with other methods.

SL-Animals consists of recordings from 60 subjects performing 19 classes of hand signals in 4 different light conditions. Subjects are non fluent, and samples only contain single signs, *i.e.* no sentences [16].

Sign-language recognition requires spatiotemporal awareness, so it is important for algorithms to be suitable for higher dimensional modeling. SL-Animals makes a great tool to experiment different models and training algorithms.

B. Pre-Processing

To natively handle AER data format, one needs to have specific neuromorphic chips. As seen in [18], there are many recent advances in neuromorphic computing, *e.g.* Loihi (Intel) [19], TrueNorth (IBM) [20], Tianjic (Tsinghua University) [21].

Without the use of dedicated chips, we can simulate the events in GPU by looping through very short time frames, at the order of milliseconds. To leverage traditional deep-learning methods, the data, then, must be converted to a compatible format for the PyTorch framework. For this, we bin these events into a histogram, using a very small time frame, to obtain a sequence of images that are then stacked into a tensor. A few reconstructed examples can be seen in Figure 2.

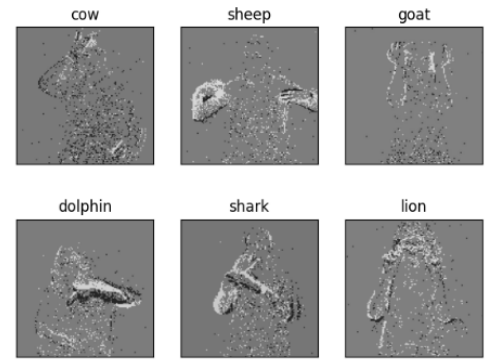


Fig. 2. Reconstructed samples of the SL-Animals dataset using a binning method.

For the number of time bins, same options can be used, like a fixed time frame between steps, that yields a variable number of images; or a fixed number of frames, yielding a variable number of time frames between the generated images. For our SFPTT results, we use a fixed number of frames to transform the events into data tensors.

C. Validation Experiments

For the network architecture, this work mainly explored topologies that had good performance on IBM's DVS Gestures [7]. Two architectures are tested:

- Model A: Conv2D(K7, C64, S1, P1) - MaxPool2D(K2, S2) - Conv2D(K3, C128, S1, P1) - MaxPool2D(K2, S2) - Conv2D(K3, C128, S1, P1) - MaxPool2D(K2, S2) - Conv2D(K3, C256, S1, P1) - MaxPool2D(K2, S2) - Dense(256) - Output(19);
- Model B: Conv2D(K7, C64, S1, P3) - MaxPool2D(K2, S2) - Conv2D(K7, C128, S1, P3) - MaxPool2D(K2, S2) - Conv2D(K3, C128, S1, P1) - MaxPool2D(K2, S2) - Conv2D(K3, C256, S1, P1) - MaxPool2D(K2, S2) - Conv2D(K3, C256, S1, P1) - MaxPool2D(K2, S2) - Conv2D(K3, C512, S1, P1) - MaxPool2D(K2, S2) - Dense(512) - Output(19),

For each model, we tested a few combinations of number of epochs, sequence length for each sample, and different learning rate schedules including Cosine Annealing (CA), which [7] reported the best results for DVS-Gesture, and

Cosine Annealing with Warm Restart (CAWR), which seems to be, from recent studies, a good similar alternative in recent studies to escape local minima [22], [23]. Our experiments use the ADAM optimizer implemented on PyTorch.

For the cross-validation, the data are split using four folds grouped by subjects. This means, a subject performing the signal ‘dog’ is always in the same fold that it is when this same subject is performing the signal ‘cat’, to prevent any type of data leakage.

STBP [12], SLAYER [13] and DECOLLE [24] results were reproduced using the best hyperparameters sets reported in [17]. Training uses a NVIDIA GTX 1080 TI 11GB GPU. A few SFPTT experiments also run on a desktop NVIDIA RTX 4060 8GB GPU.

This work focuses on cross-validation for model benchmark, with no test sets, as a similar work indicates validation results extrapolate to test sets [25].

IV. RESULTS

The result obtained on Table I confirms that SFPTT improves the sign-language recognition by an impressive margin specifically on SL-Animals, confirming the findings in [7] about the improvements on long sequences. SLAYER has a very long training time, though not leading to best performance.

TABLE I

MODEL RESULTS ON THE SL-ANIMALS DATASET.

Model	Validation Accuracy	Training Time
STBP	74.00% $\pm$ 3.15%	1 day, 6:41:29
DECOLLE	68.78% $\pm$ 0.97%	4 days, 1:20:59
SLAYER	70.26% $\pm$ 1.59%	16 days, 11:24:54
SFPTT	92.76% $\pm$ 7.36%	9:13:25

TABLE II

SFPTT EXPERIMENT RESULTS ON THE SL-ANIMALS DATASET.

Model	Epochs	Length	Scheduler	Val. Accuracy
A	100	10	CA (30)	85.24% $\pm$ 1.77%
A	200	10	CA (30)	84.96% $\pm$ 0.88%
A	200	20	CA (30)	84.30% $\pm$ 3.55%
A	200	20	CAWR (30)	92.20% $\pm$ 6.01%
A	200	40	CA (30)	77.07% $\pm$ 3.60%
A	200	20	CA (200)	92.76% $\pm$ 7.36%
B	300	10	CA (30)	82.61% $\pm$ 0.81%
B	200	15	CA (30)	86.75% $\pm$ 2.77%
B	200	20	CA (30)	88.44% $\pm$ 4.56%
B	300	20	CAWR (30)	90.60% $\pm$ 5.62%
B	200	20	CA (200)	89.47% $\pm$ 6.68%

All experiments can be seen in Table II. The best performance is achieved by Model A, after 200 epochs, with a sequence length of 20 frames and with the Cosine Annealing schedule, with 92.76% $\pm$ 7.36% accuracy across folds, as shown in Figure 3. Slightly below with 92.20% $\pm$ 6.01%, Figure 4 depicts Model A with the warm restarts.

Warm restarts seem to increase mean accuracy at the cost of higher standard deviation between folds. Figure 6 clearly

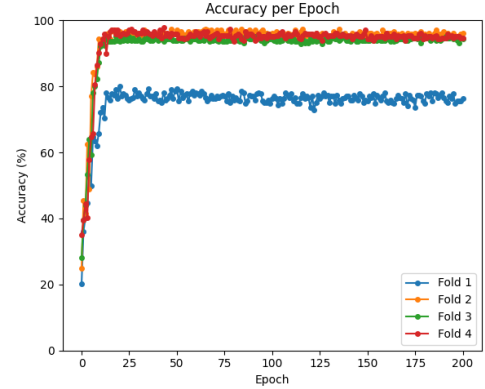


Fig. 3. Accuracy of each SFPTT validation fold for Model A using Cosine Annealing as the learning rate scheduler.

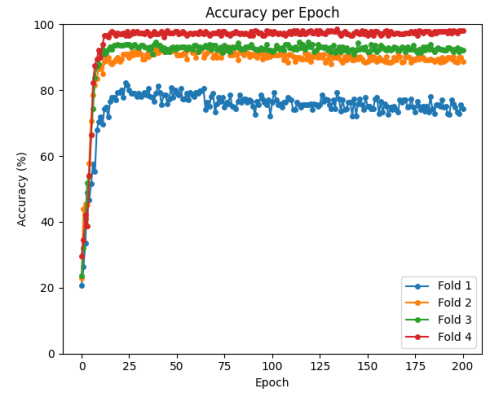


Fig. 4. Accuracy of each SFPTT validation fold for Model A using Cosine Annealing Warm Restarts as the learning rate scheduler.

shows the effect of the increased learning rate on each restart, specially in the more complex Model B, as opposed to Cosine Annealing on Figure 5. The first fold reached 100% validation accuracy, but the worse performances on the other folds reduced mean accuracy.

Regarding the number of epochs, results show that, after 200 epochs, the validation loss starts to increase. As for the sequence length, the sweet spot seems to be around 20 frames per hand-signal. The experiments with 40 lead to performance, and 10 to 15 yield just a small loss over 20.

Our intuition as to why the jump on performance happens for SFPTT, in comparison with other methods, is related to the intermediate losses that are calculated step by step, which makes more sense for the sign-language domain, since one needs the full motion in order to know which hand-signal is being performed. As discussed in Section II.C, the largest contribution to the loss, at the beginning of the hand-signal, comes from the KL divergence, which is more cohesive to this specific domain.

V. CONCLUSIONS

FPTT brings much innovation to the RNN domain, as examples, we have lower memory requirements without the need to perform backpropagation for the whole input sequence; and

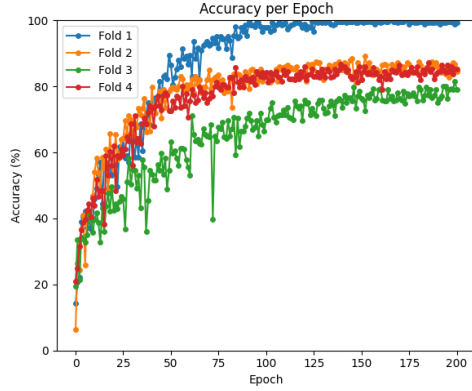


Fig. 5. Accuracy of each SFPTT validation fold for Model B using Cosine Annealing as the learning rate scheduler.

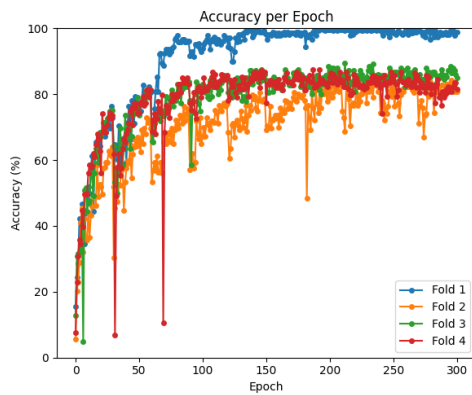


Fig. 6. Accuracy of each SFPTT validation fold for Model B using Cosine Annealing Warm Restarts as the learning rate scheduler.

clever incorporation of new terms to the loss function, to adjust training to this new SNN paradigm, such as the addition of the divergence term and moving-average weight regularizer.

The present work evaluated the use of SFPTT for sign-language recognition on the SL-Animals dataset. Our results add evidence to SFPTT being a highly effective training strategy for SNNs. It leads to superior validation accuracy when compared with previously reported SNN approaches on the same benchmark, such as STBP (74.00%), SLAYER (70.26%) and DECOLLE (68.78%). These findings support the potential of spike-based models for spatiotemporal recognition tasks, in which both motion dynamics and visual patterns must be analyzed efficiently. In addition to high predictive performance, the event-driven nature of SNNs makes them a promising alternative for low-power and real-time assistive systems.

A. Future Works

Future research may extend this study along several directions. First, further investigation of network architectures and hyperparameters may improve robustness and reduce the variance observed across folds. And, second, deployment on neuromorphic hardware and edge devices should be investigated, to validate the energy-efficiency advantages of SNNs

in real-world assistive applications, as opposed to simulating events on traditional GPUs.

REFERENCES

- [1] M. Özcan et al. *Quantifying the Energy Consumption and Carbon Emissions of LLM Inference via Simulations*. arXiv.2507.11417 (2025).
- [2] C. Posch et al. *Retinomorphic Event-Based Vision Sensors: Bioinspired Cameras With Spiking Output*. Proceedings of the IEEE. v. 102. pp. 1470-1484 (2014).
- [3] Wahid, S. Asad, A. and Mohammadi, F. *A Survey on Neuromorphic Architectures for Running Artificial Intelligence Algorithms*. Electronics. v. 13. pp. 2963 (2024).
- [4] *1st JPEG Workshop on Event-based Vision Proceedings*, ISO/IEC JTC1/SC29/WG1, wg1n100631, Online, October 24th, 2023.
- [5] J. K. Eshraghian et al. *Training Spiking Neural Networks Using Lessons From Deep Learning*. Proceedings of the IEEE, v. 111, n. 9, pp. 1016-1054 (2023).
- [6] T. Bu et al. *Optimal ANN-SNN Conversion for High-accuracy and Ultra-low-latency Spiking Neural Networks*. International Conference on Learning Representations (2023).
- [7] Yin, B., Corradi, F. & Bohtë, S.M. *Accurate online training of dynamical spiking neural networks through Forward Propagation Through Time*. Nature Machine Intelligence v. 5, pp. 518-527 (2023).
- [8] Kag, A. and Saligrama, V. *Training Recurrent Neural Networks via Forward Propagation Through Time*. Proceedings of the 38th International Conference on Machine Learning v. 139, pp. 5189-5200 (2021).
- [9] Oradovschi, B. C., *SL-Animals-DVS with sFPTT*, GitHub repository, <https://github.com/bernardoco/sl-animals-sFPTT> (2026).
- [10] Hodgkin, A. L., Huxley, A. F. *A quantitative description of membrane current and its application to conduction and excitation in nerve*. Bltn Math Biol 52, 25-71 (1990).
- [11] Neftci, E. O., Mostafa, H. and Zenke, F. *Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks*. IEEE Signal Processing Magazine v. 36, n.6, pp. 51-63 (2019).
- [12] Wu, Y. et al., *Spatio-Temporal Backpropagation for Training High-Performance Spiking Neural Networks*. Frontiers in Neuroscience v. 12-2018 (2018).
- [13] Shrestha, S. B. and Orchard, G. *SLAYER: spike layer error reassignment in time*. Proceedings of the 32nd International Conference on Neural Information Processing Systems, pp. 1419-1428 (2018).
- [14] Hasani, R. et al., *Liquid Time-constant Networks*. arXiv 2006.04439 (2020).
- [15] Amir, A. et al., *A Low Power, Fully Event-Based Gesture Recognition System*. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7388-7397 (2017).
- [16] Vasudevan, A. et al., *SL-Animals-DVS: event-driven sign language animals dataset*. Springer-Verlag v. 25, n. 3, pp. 505-520 (2022).
- [17] Schechter, R. C., and Gomes, J. G. R. C. *Enhancing Gesture Recognition Performance Using Optimized Event-Based Data Sample Lengths and Crops*. 2024 IEEE 15th Latin America Symposium on Circuits and Systems (LASCAS), pp. 1-5 (2024).
- [18] Xiao, Y. et al., *Recent Progress in Neuromorphic Computing from Memristive Devices to Neuromorphic Chips*. Advanced Devices & Instrumentation v. 5, pp. 0044 (2024).
- [19] M. Davies et al., *Loihi: A Neuromorphic Manycore Processor with On-Chip Learning*, IEEE Micro, vol. 38, no. 1, pp. 82-99 (2018).
- [20] F. Akopyan et al., *TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, no. 10, pp. 1537-1557 (2015).
- [21] Pei, J., Deng, L., Song, S. et al., *Towards artificial general intelligence with hybrid Tianjic chip architecture*. Nature 572, pp. 106-111 (2019).
- [22] Loshchilov, I. and Hutter, F. *SGDR: Stochastic Gradient Descent with Warm Restarts*. arXiv, 1608.03983 (2017).
- [23] Varshney, A. K. *When to restart? Exploring escalating restarts on convergence*. Workshop on Scientific Methods for Understanding Deep Learning (2026).
- [24] Kaiser, J., Mostafa, H. and Neftci, E. *Synaptic Plasticity Dynamics for Deep Continuous Local Learning (DECOLLE)*. Frontiers in Neuroscience v. 14-2020 (2020).
- [25] Schechter, C. R. *Enhancing Gesture Recognition Performance Using Optimized Event-Based Data and Spiking Neural Networks*. Universidade Federal do Rio de Janeiro, M. Sc. dissertation (2024).