Investigation of Enhancement Strategies for Recurrent Spiking Neural Network based Brain-Machine Interface Decoding

Wilson Tansil*, Nur Ahmadi *†‡, Timothy G. Constandinou‡§ and Dessi Puji Lestari*†
*School of Electrical Engineering and Informatics, Bandung Institute of Technology, Indonesia
†Center for Artificial Intelligence, Bandung Institute of Technology, Indonesia
‡Department of Electrical and Electronic Engineering, Imperial College London, London, SW7 2AZ, UK
§UK Dementia Research Institute, Care Research & Technology Centre at Imperial College London & University of Surrey
E-mail: wilsontansil@gmail.com, nahmadi@itb.ac.id, t.constandinou@imperial.ac.uk, dessipuji@itb.ac.id

***Abstract*—Brain-Machine Interfaces (BMIs) for motor function restoration face a key challenge in decoding complex neural signals. While Recurrent Spiking Neural Networks (RSNNs) are promising for this task, they often suffer from limited temporal representation, high computational loads, and training instability. This research investigates three strategies to enhance RSNNs: (1) learnable synaptic delays using a 1D Dilated Convolution (DCLS1D) layer to enrich temporal modeling, (2) linear interpolation to reduce synaptic operations, and (3) Temporal Efficient Training (TET) to improve training stability. The models were evaluated on the NeuroBench Non-Human Primate (NHP) Motor Prediction task. Results indicate that learnable delays did not improve performance. Linear interpolation reduced computational cost but compromised accuracy on non-linear data. In contrast, the TET approach consistently improved both model accuracy (measured by R^2 and RMSE) and training stability, demonstrating its effectiveness in overcoming key RSNN limitations.**

I. Introduction

Brain-Machine Interfaces (BMIs) have become a rapidly growing research focus, with primary applications in healthcare, prosthetics, and the control of external devices [1], [2]. A major challenge in BMI development is handling complex neural signals, high power consumption, and the thermal impact of devices, especially for implanted ones. Neural data generated by a large number of electrodes require algorithms for real-time processing without sacrificing power efficiency, model accuracy, and the safety of the device for brain tissue.

To address challenges in BMIs, this research focuses on optimizing the neuromorphic approach using Spiking Neural Networks (SNNs). Inspired by the brain's efficiency, SNNs utilize spike-based processing for fast, low-power computation, making them ideal for safely analyzing neural signals in real-time [3], [4]. This study will optimize an SNN for the Non-human Primate (NHP) Motor Prediction task from the NeuroBench benchmark [5]. The goal is to accurately predict the two-dimensional velocity of finger movements from neural recordings of the sensorimotor cortex.

In BMI applications, decoding neural signals to predict motor activity is a significant challenge due to the complex, temporal, and noisy nature of the signals. Traditional approaches like the Artificial Neural Network (ANN) are less efficient for spike-based data that mimics biological activity. As a solution, [6] proposed a Recurrent Spiking Neural Network (RSNN) model, which proved superior in accuracy and efficiency. Their tinyRSNN design is a strong candidate for implementation in implantable BMIs, yet there is still room for improvement in terms of temporal representation, synaptic operation efficiency, and training stability.

Temporal representation is a critical aspect because information in the biological nervous system is encoded not only by which neurons are active but also by the precise timing of their spikes. This temporal precision is a key element in encoding complex motor activities [7]. Spiking neurons fundamentally act as coincidence detectors [8], responding more strongly to input spikes that arrive simultaneously than to those that are asynchronous [9]. Therefore, the model must recognize not only spatial activation patterns but also their precise temporal dynamics. Synaptic delay plays a vital role here, as delays in spike transmission allow important information to be amplified through the superposition of previous spikes. Furthermore, heterogeneous delays enable neurons to detect more complex spatiotemporal patterns beyond simple synchronization. [6] also noted that while their RSNN performed exceptionally, its ability to capture complex temporal dynamics could be enhanced, with synaptic delay being a potential strategy to address this.

Furthermore, synaptic operation efficiency is a fundamental challenge in implementing RSNN models due to their extremely high computational load. This demands that neural decoding algorithms be not only accurate but also resource-efficient. In the model implemented by [6], the synaptic operation metric within the NeuroBench benchmark is a key component for assessing efficiency and is highly dependent on the model's synaptic activity rate. [10] also demonstrated that processing each time step individually is often inefficient and leads to excessive computational overhead. In reality, motor movements can often be approximated by a linear expression defined by a few key points, indicating that crucial information is concentrated at specific moments, while neural activity

979-8-3315-7206-8/25/$31.00

between these moments can be interpolated.

Finally, training stability is a significant concern in RSNN development. The standard training method for SNN models, Standard Direct Training, used by [6], calculates the loss based on the average output over the entire time duration. As identified by [11], this approach can cause the training process to get stuck in sharp local minima, which harms the model's ability to generalize to new data. Optimizing based on the average output O(t) can also mask strong error signals at specific moments with good performance at other time steps, reducing the model's ability to correct time-specific errors. To improve training stability and efficiency, Temporal Efficient Training (TET) can be applied. TET helps avoid generalization issues (sharp local minima) common in training long temporal models and enhances efficiency by calculating the loss incrementally at each time step [11].

II. Methodology

A. Temporal Representation with Learnable Synaptic Delays

Neural information is encoded not just by which neurons are active, but precisely when they fire. Spiking neurons fundamentally act as coincidence detectors, responding more strongly to synchronous inputs [9]. Incorporating heterogeneous synaptic delays allows a network to detect complex spatiotemporal patterns beyond simple synchronicity. Recognizing this, [6] suggested that adding synaptic delays was a promising strategy to improve the temporal dynamics of their RSNN model.

To implement this, we enrich the model's temporal representation by replacing the temporally static linear layer of the baseline RSNN with a 1D Dilated Convolution with Learnable Spacings (DCLS1D) layer, adapted from [9]. This layer transforms the connections into dynamic temporal filters.

1) DCLS1D Mechanism and Implementation: The DCLS1D layer models each synaptic connection as a sparse 1D temporal convolution. To ensure the delays are differentiable and can be learned via backpropagation, each connection is modeled as a Gaussian kernel. The center of the Gaussian represents the learnable delay (d_{ij}), while the area under its curve represents the learnable synaptic weight (w_{ij}).

The input current $I_i[t]$ for a neuron i is the sum of convolutions between input spike trains S_j from each presynaptic neuron j and their respective kernels k_{ij}:

$$I_i[t] = \sum_j (k_{ij} * S_j)[t] \tag{1}$$

Prior to this operation, causal left padding is applied to the input sequence to prevent acausal information flow. Optional right padding is also used to enhance the model's expressive power, allowing it to learn dependencies that extend beyond the observed sequence length [9].

To maintain recurrence efficiently, the final current $I_{\text{final}}(t)$ fed to the neuron's membrane potential update is the sum of the DCLS layer's output at the current and previous timesteps:

$$I_{\text{final}}(t) = I_{\text{DCLS}}(t) + I_{\text{DCLS}}(t-1) \tag{2}$$

2) Parameter Initialization: For the DCLS1D layer, we use Kaiming Uniform initialization [12], following the successful implementation by [9]. Fluctuation-based initialization methods, while effective for standard convolutional SNNs [13], are unsuitable here. This is because DCLS1D decouples the layer's parameters into distinct components (weight, delay, and standard deviation of the Gaussian kernel), which cannot be collectively optimized by a single initialization scheme designed for standard convolutional kernels.

B. Computational Efficiency via Keypoint Interpolation

A fundamental challenge in implementing Recurrent Spiking Neural Networks (RSNN) for BMI applications is their significant computational load, driven primarily by the high volume of synaptic operations [6]. This demands decoding algorithms that are not only accurate but also highly resource-efficient. Inspired by the observation that processing every time step is often redundant and that motor trajectories can be effectively approximated linearly between key moments [10], we propose a solution to reduce this computational burden. The validity of this approach is supported by findings that linear interpolation results in negligible error while achieving high accuracy ($R^2 = 0.988$), making it a viable strategy for SNNs.

Our solutions, built upon the baseline RSNN model, restrict full spike propagation to specific keypoints within a time interval and use linear interpolation for the intermediate steps. We explore two distinct approaches: input-oriented and output-oriented interpolation.

Input-Oriented Interpolation: In this approach, we interpolate the presynaptic input to the neurons. Instead of performing full spike propagation at every time step, the mechanism is as follows:

1) At the start of an interval, a full propagation is executed to establish an initial state, which serves as the lower bound, v_{low}.
2) At the end of the interval, a second full propagation is performed to determine the upper bound, v_{high}.
3) For all time steps between these two keypoints, the input to the LIF neurons is not computed via synaptic operations. Instead, it is efficiently estimated using linear interpolation between v_{low} and v_{high}. This step ensures that synapses are only activated at specific, critical moments.

Output-Oriented Interpolation: This second approach interpolates the final output of the model rather than its internal inputs. This is based on the finding that for motor movements, only a few important time steps need to be fully processed, while the rest can be predicted [10]. The mechanism is simplified:

1) The full RSNN model is executed only at key-time steps (e.g., at the start and end of an interval).
2) This yields predicted velocity outputs, v_{low} and v_{high}, at these keypoints.
3) The velocity predictions for all intermediate time steps are then calculated by linearly interpolating between these two output values.

In cases where the total number of time steps is not perfectly divisible by the keypoint interval, a final iteration is performed on the remaining steps to ensure the entire sequence is processed. The primary objective of this design is to prevent the continuous activation of synaptic connections, thereby drastically reducing the total number of synaptic operations and significantly improving the model's computational efficiency.

C. Training Stability with Temporal Efficient Training (TET)

A significant challenge in training Recurrent Spiking Neural Networks (RSNN) is ensuring model stability and generalization. Standard methods like Standard Direct Training (SDT), utilized by [6], calculate a single loss value based on the average network output over an entire time sequence, which can be represented as [11]:

$$\mathcal{L}_{\text{SDT}} = \mathcal{L}\left(\frac{1}{T}\sum_{t=1}^{T} O(t), y\right) \tag{3}$$

where T is the total number of time steps.

To address the limitations of SDT, we implement the Temporal Efficient Training (TET) approach [11]. The core concept of TET is to shift from calculating a single loss on the averaged output to calculating the loss at each individual time step and then averaging these loss values.

$$\mathcal{L}_{\text{TET}} = \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}(O(t), y) \tag{4}$$

This method effectively re-weights the gradient contribution from each moment in time. Since the error at each time step is unlikely to be zero, the optimizer is forced to continuously adjust the weights to improve accuracy across the entire sequence, not just its average. This encourages the training process to seek flat minima, which are theoretically proven to offer better generalization. As proven by Deng et al. [11], $\mathcal{L}_{\text{TET}}$ is an upper bound for the standard loss, meaning that minimizing TET loss also helps minimize SDT loss.

Furthermore, to enhance stability and mitigate the impact of outlier outputs, a regularization penalty is added. The final loss function is a weighted combination of the TET loss and a Mean Squared Error (MSE) loss:

$$\mathcal{L}_{\text{total}} = (1-\alpha)\mathcal{L}_{\text{TET}} + \alpha\mathcal{L}_{\text{MSE}} \tag{5}$$

where α is a hyperparameter that controls the regularization strength. It is important to note that this modified loss function is applied only during the training phase. During inference, the model's standard evaluation rules are used to ensure a fair comparison.

D. Training and Evaluation Methods

The training protocol replicates the pre-training procedure established by [6]. In this phase, the models are trained on the complete set of available 'Indy' datasets [14], which consists of multichannel electrophysiological recordings from the sensorimotor cortex of a non-human primate reaching task. The input to our model is a binned spike train (4ms) from cortical electrodes, and the target output is corresponding 2D velocity of primate's finger movement. The process concludes after this pre-training stage, where an early stopping mechanism is employed to identify and save the best-performing model checkpoint for evaluation.

To assess the proposed solutions, we employ a comprehensive suite of evaluation metrics. The primary metrics are derived from the NeuroBench framework, which include memory footprint, the coefficient of determination (R^2), activation sparsity, and synaptic operations. In addition, the evaluation is supplemented with standard statistical metrics to precisely measure prediction accuracy against the ground truth values: Root Mean Squared Error (RMSE) and the Pearson Correlation Coefficient. The evaluation is conducted on a separate test dataset comprising three neural recordings of varying durations to assess the model's generalization and ensure performance reflects real-world scenarios.

III. Results

In this section, the implemented models are evaluated and compared using three distinct recordings: `indy_20170131_02.mat`, `indy_20160622_01.mat`, and `indy_20160630_01.mat` from [14]. The evaluation utilizes metrics from the NeuroBench benchmark [5] (footprint, R^2, activation sparsity, and synaptic operations), supplemented with Root Mean Squared Error (RMSE) and the mean Pearson Correlation Coefficient (μCorr). The detailed results are presented in Table I.

A. Experimental Setup and Baseline Model

All proposed solutions were developed and compared against a baseline model [6]. The baseline architecture consists of an input layer, a hidden layer of 64 Leaky Integrate-and-Fire (LIF) neurons, and a readout layer of 2 non-spiking Leaky Integrator (LI) neurons, corresponding to the X and Y velocity outputs. The model was trained for a maximum of 200 epochs with an early stopping patience of 10.

B. Experiment 1: Learnable Synaptic Delays

In this experiment, the static linear connections of the baseline model were replaced with the DCLS1D layer. The maximum delay was set to 100 and 50 ms (25 and 13 time steps), and the Kaiming Uniform initializer was used. A key modification was the use of replicate padding (161 epoch) instead of zero-padding (38 epoch). Because the input data is binary, zero-padding creates ambiguity between actual non-spikes (value 0) and padded values. Replicate padding resolves this by using the data's edge values for padding, leading to improved training performance (Figure 1). After experimenting with various learning rates for the positional delays, no significant performance difference was observed (Figure 2), so a rate of 0.1 was chosen. For this experiment, the hyperparameters for the synaptic delay model are listed in Table II.

Compared to the baseline, the synaptic delay model exhibited a rapid performance increase in the first 20 epochs

TABLE I
AVERAGE ACCURACY AND EFFICIENCY METRICS ON THE TESTED RECORDING.

Model	Accuracy Metrics			Efficiency Metrics			
	R^2	RMSE	μCorr	Footprint	Act. Spars.	Eff. AC	Dense
Baseline	0.6117	0.1569	0.7801	**21,000**	0.9842	419.17	10,368.0
Baseline + TET	**0.6167**	**0.1561**	**0.7830**	**21,000**	0.9844	418.33	10,368.0
Baseline + Delay 0.1s (replicate) + TET	0.5969	0.1597	0.7671	61,960	0.9864	64,669.45	3,328,607.65
Baseline + Delay 0.1s (replicate)	0.5903	0.1608	0.7626	61,960	0.9865	64,664.36	3,328,607.65
Baseline + Delay 0.05s (replicate)	0.5900	0.1611	0.7611	61,960	0.9860	29,799.84	932,035.15
Baseline + Delay 0.1 (zeros)	0.5239	0.1737	0.7222	61,960	0.9834	5,307.75	3,328,607.65
Baseline + Output-Oriented Linear Interpolation (2 keys)	0.4859	0.1812	0.6883	**21,000**	0.9710	237.71	5,184.01
Baseline + Input-Oriented Linear Interpolation (2 keys)	0.4176	0.1946	0.6360	**21,000**	0.9857	205.98	5,184.11
Baseline + Output-Oriented Linear Interpolation (8 keys)	0.1788	0.2307	0.3988	**21,000**	0.8835	105.43	**1,296.17**
Baseline + Input-Oriented Linear Interpolation (8 keys)	0.1194	0.2408	0.3352	**21,000**	**0.9891**	**50.04**	**1,296.17**

TABLE II
EXPERIMENT HYPERPARAMETER FOR LEARNABLE SYNAPTIC DELAYS APPROACH.

Hyperparameter	Value
Position Delay Learning Rate	0.1, 0.01, 0.002
Maximum Delay	0.1, 0.005 second
Weight Initialization	Kaiming Uniform
Padding Mode	Replicate and Zeros
Kaiming Uniform	(-maxdelay // 2, maxdelay // 2)
Left Padding	maxdelay - 1
Right Padding	maxdelay // 2

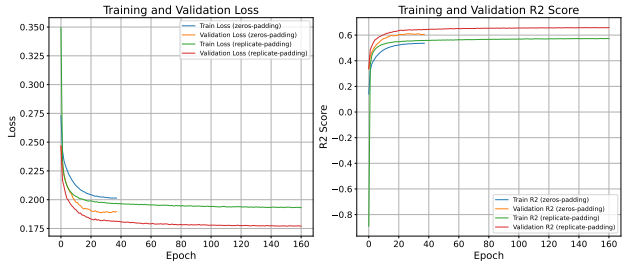

Fig. 1. Comparison of training performance using replicate padding vs. zero padding.

but stagnated thereafter, while the baseline showed slower but more consistent improvement, converging more efficiently (Figure 3). Despite the increased architectural complexity, the synaptic delay models' accuracy was inconsistent and often inferior to the baseline. As shown in Table I, they also had a substantially larger memory footprint (∼3x) and orders of magnitude more synaptic operations, indicating significantly lower computational efficiency. These results suggest that enriching temporal dynamics through learnable delays does not guarantee better performance and may require more advanced optimization to justify the added computational cost.

C. Experiment 2: Linear Interpolation

This experiment tested two variants of linear interpolation, input-oriented and output-oriented, with keypoint distances of 2 and 8. Although the output-oriented approach consistently

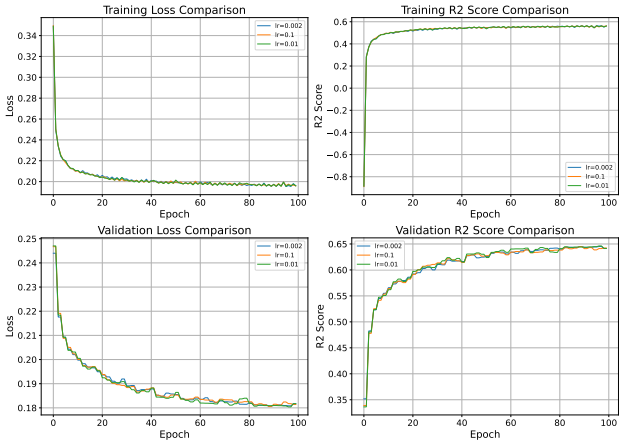

Fig. 2. Training results with varying learning rates for positional delay.

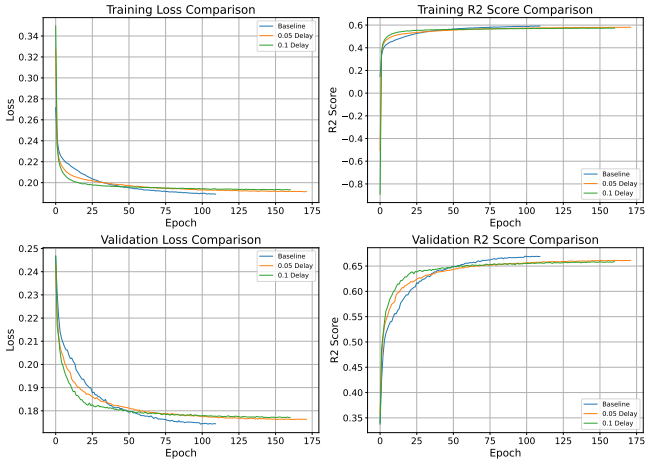

Fig. 3. Training performance comparison: Baseline vs. Synaptic Delay model.

performed better than the input-oriented one (Figure 4), both methods ultimately underperformed the baseline model (Figure 5). This failure can be attributed to a fundamental mismatch between the linear assumption of interpolation and the non-

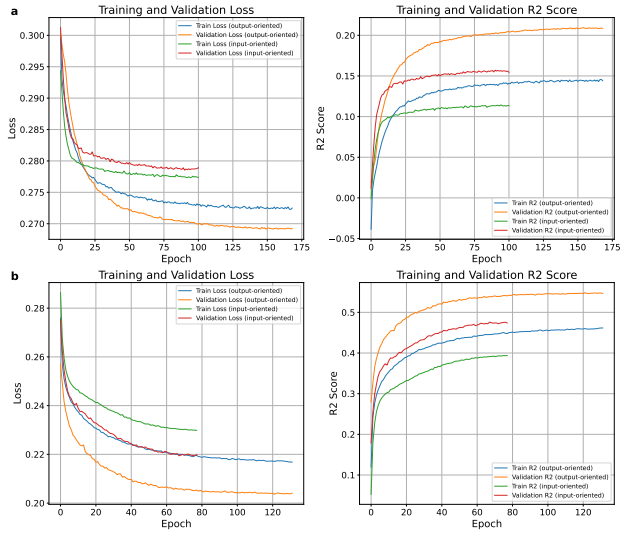

Fig. 4. Training performance comparison: (a) 8 keys and (b) 2 keys Input-Oriented vs Output-Oriented Linear Interpolation models.

linear nature of the problem, albeit at different stages for each approach.

For the input-oriented method, the failure occurred at the neuron level. The model's internal post-synaptic dynamics are highly non-linear, as shown in Figure 6. By assuming a predictable, straight-line behavior for neuron inputs, this approach effectively ignores critical non-linear events occurring between keypoints.

For the output-oriented method, the failure occurred at the final prediction stage. While the model could follow the general velocity patterns of the labels, the interpolation process overly smoothed the final output, as shown in Figure 7. This caused the model to lose the sharp, detailed changes in velocity that are characteristic of the actual finger movements.

The unsuitability of this approach is confirmed by the low R^2 scores and high RMSE values across all recordings (Table I). The performance degradation was most severe for the 8-keypoint models, whose low R^2 scores indicate they captured very little of the meaningful relationship in the data.

D. Experiment 3: Temporal Efficient Training (TET)

To address training stability, we applied the TET loss function to both the baseline and synaptic delay models. The results consistently demonstrated superior performance. The baseline model with TET achieved a lower loss and a higher R^2 score than the standard training (Figure 8). Similarly, the synaptic delay model's performance was significantly enhanced by TET (Figure 9). This confirms that TET effectively guides the optimization towards flatter minima, avoiding sharp local optima. While this extended the training duration slightly (e.g., from 110 to 133 epochs for the baseline), it led to a more stable process and a more accurate final model.

Across all recordings, the baseline model, particularly when combined with TET, demonstrated the best overall performance in terms of accuracy (R^2, RMSE, μCorr) and memory

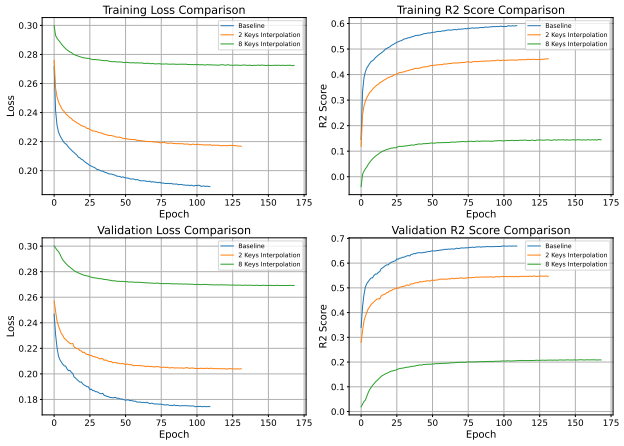

Fig. 5. Training performance comparison: Baseline vs. Output-Oriented Linear Interpolation models.

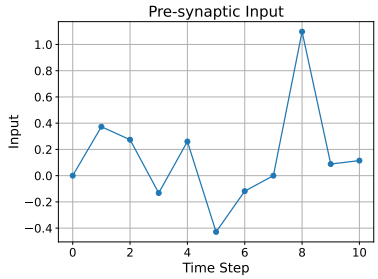

Fig. 6. Illustration of the highly non-linear behavior of presynaptic input within an interpolation interval.

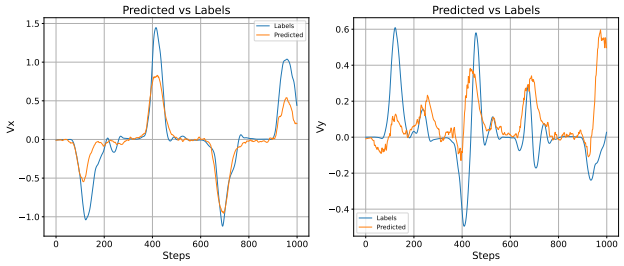

Fig. 7. Comparison of the predicted velocity from the output-oriented model against the ground truth labels.

footprint. As detailed in Table I, the baseline with TET achieved a top R^2 score of 0.6167 and a μCorr of 0.7830. This confirms that the enhanced training stability from TET translates into a more robust and slightly more accurate final model, establishing this configuration as the most effective.

IV. Conclusions

This study evaluated three enhancements to a baseline RSNN for motor decoding: learnable synaptic delays, linear interpolation, and Temporal Efficient Training (TET). The results show that architectural modifications aimed at improving temporal complexity or efficiency were ineffective. Learnable delays significantly increased computational cost

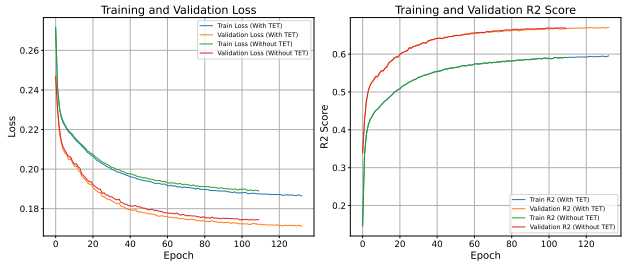

Fig. 8. Baseline model training with and without TET.

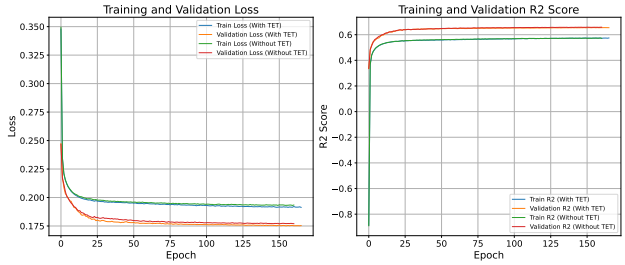

Fig. 9. Synaptic Delay model training with and without TET.

without providing an accuracy benefit, while linear interpolation failed due to its unsuitability for capturing the patterns within the data. In contrast, Temporal Efficient Training (TET) successfully improved training stability, making it the most reliable approach tested. While this only yielded a marginal accuracy increase (a 0.05 gain in the R^2 score), it shows that optimizing the training process is more practical than increasing architectural complexity for a minimal return.

Acknowledgment

This research was funded by the Directorate of Research and Innovation, Institut Teknologi Bandung (DRI ITB) under ITB Research Program (Grant No. 841/IT1.B07.1/TA.00/2025). The authors also gratefully acknowledge the use of the Chameleon Cloud testbed [15] for the computational experiments presented in this paper.

References

[1] N. Ahmadi, T. Adiono, A. Purwarianti, T. G. Constandinou, and C.-S. Bouganis, "Improved spike-based brain-machine interface using bayesian adaptive kernel smoother and deep learning," *IEEE Access*, vol. 10, pp. 29 341–29 356, 2022.

[2] N. Ahmadi, T. G. Constandinou, and C.-S. Bouganis, "End-to-end hand kinematic decoding from lfps using temporal convolutional network," in *2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, IEEE, 2019, pp. 1–4.

[3] G. Indiveri and R. Douglas, "Neuromorphic cognition," in *Encyclopedia of Computational Neuroscience*, D. Jaeger and R. Jung, Eds., 2nd, Springer, 2022.

[4] C. D. Schuman, S. R. Kulkarni, M. Parsa, J. P. Mitchell, P. Date, and B. Kay, "Opportunities for neuromorphic computing algorithms and applications," *Nature Computational Science*, vol. 2, no. 1, pp. 10–19, 2022.

[5] J. Yik, K. Van den Berghe, D. den Blanken, *et al.*, "The neurobench framework for benchmarking neuromorphic computing algorithms and systems," *Nature Communications*, vol. 16, no. 1, p. 1545, 2025.

[6] T. Liu, J. Gygax, J. Rossbroich, Y. Chua, S. Zhang, and F. Zenke, "Decoding finger velocity from cortical spike trains with recurrent spiking neural networks," in *2024 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, IEEE, 2024, pp. 1–5.

[7] S. D'Agostino, F. Moro, T. Torchet, *et al.*, "Denram: Neuromorphic dendritic architecture with rram for efficient temporal processing with delays," *Nature Communications*, 2024.

[8] C. Rossant, S. Leijon, A. K. Magnusson, and R. Brette, "Sensitivity of noisy neurons to coincident inputs," *Journal of Neuroscience*, vol. 31, no. 47, pp. 17 193–17 206, 2011.

[9] I. Hammouamri, I. K. Hassani, and T. Masquelier, "Learning delays in spiking neural networks using dilated convolutions with learnable spacings," in *2024 International Conference on Learning Representations (ICLR)*, 2024.

[10] A. Vasilache, J. Krausse, K. Knobloch, and J. Becker, "Hybrid spiking neural networks for low-power intracortical brain-machine interfaces," in *2024 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, IEEE, 2024, pp. 1–5.

[11] S. Deng, Y. Li, S. Zhang, and S. Gu, "Temporal efficient training of spiking neural network via gradient reweighting," in *2022 International Conference on Learning Representations (ICLR)*, 2022.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[13] 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, 2022.

[14] J. E. O'Doherty, M. M. B. Cardoso, J. G. Makin, and P. N. Sabes, *Nonhuman primate reaching with multichannel sensorimotor cortex electrophysiology (version 1)*, Zenodo, [Dataset], May 2017. DOI: 10.5281/zenodo.583331.

[15] K. Keahey, J. Anderson, Z. Zhen, *et al.*, "Lessons learned from the chameleon testbed," in *2020 USENIX annual technical conference (USENIX ATC 20)*, 2020, pp. 219–233.