Spike-Transmission Delays Improve the Accuracy–Efficiency Trade-Off for Linear Readouts of Spiking Populations

Alina Lytovchenko(✉), Nicolas Martalog, and Jeff Orchard

Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada
alytovch@uwaterloo.ca

Abstract. Linear decoders for spiking populations typically use only the instantaneous filtered activity of each neuron and therefore do not fully exploit the temporal structure present in neural responses. We study a delay-augmented linear readout in which each neuron contributes several time-shifted copies of its activity, giving the decoder access to recent response history without introducing recurrence or nonlinear processing. To ensure a fair comparison, we keep the total number of decoder inputs fixed across conditions: when more delay taps are added per neuron, the number of neurons is reduced accordingly. On one-dimensional reconstruction of band-limited white-noise signals, we compare this approach with a baseline readout that uses current activity alone in populations of leaky integrate-and-fire (LIF) and adaptive LIF (ALIF) neurons. Delay-augmented decoding improves accuracy under matched decoder dimensionality, with gains that depend on signal timescale and firing regime. The largest improvements occur for ALIF populations and reach nearly 40% relative MSE reduction in the best settings, while LIF populations show smaller and more regime-dependent benefits. These results show that fixed spike-transmission delays can improve decoding efficiency without recurrence or learned temporal dynamics.

Keywords: Spiking Neural Networks · Temporal Coding · Neural Adaptation · Energy-Efficient Computing

1 Introduction

Spiking neural networks (SNNs) are widely explored as brain-inspired models that communicate through discrete spike events, enabling sparse, event-driven computation suitable for neuromorphic hardware [4,10,15,16]. However, achieving good task performance often requires a large number of neurons or high firing rates, reducing overall efficiency. This brings the main question into focus: how efficiently can signals be reconstructed from spiking activity?

A common approach is linear decoding from filtered population activity [6, 18,25]. While simple and interpretable, standard feedforward decoders typically use only instantaneous neural activity. The temporal filtering introduced by

L. Pasa et al. (Eds.): ICANN 2026, LNCS 17089, pp. 272–283, 2027.
https://doi.org/10.1007/978-3-032-38398-3_23

neural and synaptic dynamics can make the desired signal difficult to recover from instantaneous activity alone.

Previous work addresses this limitation by embedding temporal structure directly into the network. Recurrent and reservoir models transform recent history into richer internal states [11,12]. Statistical decoders explicitly incorporate spike history or latent temporal structure [14,21,26], while other approaches introduce learnable delays inside the network itself [5,9,19,20,24,27]. Although effective, these methods modify the encoding network or add complexity to optimization, leaving open a narrower question: how much can decoding accuracy improve if explicit temporal context is provided only through fixed delays between the neurons and the readout?

This paper explores that question using per-neuron spike-transmission delays applied to the spike trains of a fixed feedforward spiking population. To ensure a fair comparison, we match the total number of inputs to the linear decoder across conditions. Because delay taps yield multiple inputs to the readout node per neuron, the delayed setting uses fewer neurons so that the overall decoder size remains the same. This lets us test whether temporal context helps beyond simply increasing the number of decoder features. We perform this matched-dimensionality comparison on continuous signal reconstruction tasks with leaky integrate-and-fire (LIF) and adaptive LIF (ALIF) neuron populations. Adaptation in ALIF neurons creates temporal dependencies in spike responses [3,13,17], potentially increasing the benefits of explicit spike-transmission delays.

Thus, the primary contribution of this paper is a controlled evaluation of fixed per-neuron delays inserted between a spiking population and a linear readout. Specifically, we introduce a matched-dimensionality comparison to isolate the effect of these delays and show that using delayed copies of neural activity in the readout can improve decoding accuracy under matched decoder dimensionality, especially for adaptive neuron populations.

2 Related Work

2.1 Linear Decoding from Spiking Populations

Linear decoding from neural populations is a long-established idea. Classical work on population codes showed that continuous variables can often be reconstructed accurately by linear estimators applied to nonlinear neural responses [18,25]. In computational neuroscience, this perspective was developed further in the Neural Engineering Framework (NEF), where low-dimensional variables are represented by spiking populations and recovered through linear decoders [6]. Our baseline readout belongs to this family: a feedforward spiking population encodes a low-dimensional signal, and a linear decoder reconstructs it from filtered population activity.

2.2 Temporal Decoding from Spike Trains

Another approach is to decode signals directly from spike trains using statistical or Bayesian models. Point-process and generalized linear model (GLM)

methods include spike history, population history, and external variables [14,21]. State-space methods such as Kalman filters are also widely used for continuous decoding, especially in motor tasks [26]. For leaky integrate-and-fire populations, Bayesian decoders have also been derived directly from spiking neuron models [7,8]. These methods are often more flexible than a simple linear readout, but they also depend more strongly on the choice of model. Our goal is simpler: we ask how much temporal information can be recovered by adding fixed delay taps to a linear decoder.

2.3 Temporal Processing Through Recurrence and Adaptation

Temporal processing in spiking networks is more commonly achieved through recurrence. Liquid state machines and related reservoir approaches use recurrent spiking dynamics to map recent input history into a high-dimensional state from which a linear readout is trained [11,12,23]. In contrast, our encoder remains feedforward, as we do not rely on recurrent state, learned recurrent dynamics, or memory stored in the network connectivity itself.

Another relevant line of work studies intrinsic neuronal mechanisms for temporal computation. Adapting spiking neurons have been shown to support temporally extended computations more effectively than non-adapting neurons, both in recurrent spiking models and in population-level analyses of coding and decoding [3,13,17]. This literature motivates our comparison between LIF and ALIF populations. If adaptation makes the neural response depend on recent spike history, then a decoding strategy with access to delayed copies of that activity should be better positioned to extract the encoded signal.

2.4 Delays in Spiking Neural Networks

The closest related work involves temporal processing using explicit delays. Voelker et al. [22] showed that accurate delays and richer synaptic dynamics can support delay-line-like computations in spiking networks. In the broader SNN literature, many methods focus on learning delays inside the network. For example, SLAYER introduced gradient-based training of SNNs with axonal delays [19], and later work learned delays using differentiable spike representations [24] or joint delay-weight plasticity [27]. More recent approaches learn axonal delays for temporal tasks [20], use dilated temporal structures [9], or jointly learn delays, weights, and adaptation [5].

In contrast, we do not learn delays or use recurrence. Instead, we study a fixed feedforward spiking population with a readout-side fixed delay bank and a linear decoder. Our goal is to test whether this simple design improves decoding under a fixed decoder size. Overall, our work provides a controlled study of how adding a small amount of temporal context at the readout affects the accuracy–efficiency trade-off of linear decoding.

3 Methods

3.1 Overview

We encode a time-varying scalar signal $x(t)$ into a population of N spiking neurons and attempt to recover it with a linear readout. Each neuron n receives the input signal as a driving current and produces a spike train $S_n(t)$. In the baseline setting, the readout is computed from the instantaneous filtered spike trains of the population. In the delayed setting, each neuron contributes multiple time-shifted copies of its spike train, forming a delay bank. The spikes synapse onto the readout node, where a synaptic weight and filter are applied. Finally, all inputs to the readout node are added together to form $\hat{x}(t)$, an estimate of the desired output. The connection weights of this linear decoder are learned on a separate training segment.

3.2 Neural Models

Leaky Integrate-and-Fire (LIF). Each LIF neuron maintains a membrane potential $v_n(t)$ that integrates its input current $J_n(t)$ according to

$$\tau_m \, \dot{v}_n = -v_n + J_n(t), \tag{1}$$

where τ_m is the membrane time constant.

Whenever v_n reaches a threshold v_{th}, a spike is emitted and v_n is reset to zero. The neuron resumes integrating after a refractory period τ_{ref}.

Adaptive LIF (ALIF). The ALIF neuron extends the LIF dynamics with a spike-triggered adaptation current $q_n(t)$:

$$\tau_m \, \dot{v}_n = -v_n - q_n + J_n(t), \tag{2}$$

$$\tau_a \, \dot{q}_n = -q_n + \beta \, S_n(t) \tag{3}$$

where τ_a is the adaptation time constant and β is the adaptation strength. Because q_n builds up after each spike and decays slowly, it suppresses neuronal excitability over time and reduces the firing rate even under constant input. As a result, the relationship between the neuron's input and spike response becomes history-dependent, so fixed decoders trained based on stationary assumptions may not work well unless adaptation is explicitly accounted for.

3.3 Spike Trains and Delay Taps

The spike train, $S_n(t)$, of neuron n is the sum of Dirac delta functions at its spike times $\{t_{n,k}\}$,

$$S_n(t) = \sum_k \delta(t - t_{n,k}), \tag{4}$$

where $t_{n,k}$ is the time of its kth spike. In the baseline setting, each neuron contributes a single temporally filtered activity trace obtained by convolving its spike train with a causal exponential synaptic filter,

$$a_n(t) = (h_{\tau_s} * S_n)(t), \tag{5}$$

where $h_{\tau_s}(t)$ is a causal exponential synaptic filter with time constant τ_s.

In the delay-augmented setting, each neuron contributes M tapped copies of its activity with delays $\{d_0, d_1, \ldots, d_{M-1}\}$, where $d_0 = 0$ corresponds to the instantaneous channel. The mth feature of neuron n is

$$a_n^{(m)}(t) = (h_{\tau_s} * S_n)(t - d_m), \tag{6}$$

with $a_n^{(m)}(t) = 0$ for $t < d_m$. In implementation, these taps are realized as explicit per-neuron spike-transmission delays applied before a shared readout synapse. For a linear time-invariant synaptic filter, this gives the feature representation above.

For each neuron n, we collect the M delayed activity channels into the row vector

$$\mathbf{a}_n(t) = \left[a_n^{(0)}(t), \ldots, a_n^{(M-1)}(t)\right] \in \mathbb{R}^{1\times M}. \tag{7}$$

The complete delayed feature vector from the entire population is then

$$\mathbf{z}(t) = \left[\mathbf{a}_1(t) | \ldots | \mathbf{a}_N(t)\right] \in \mathbb{R}^{1\times NM}. \tag{8}$$

Thus, the instantaneous decoder uses one feature per neuron, whereas the delayed decoder uses M features per neuron. The delay bank introduces temporal context into the readout while keeping the decoder itself linear and feedforward.

3.4 Linear Readout and Decoder Training

Although the readout is linear in the feature vector $\mathbf{z}(t)$, it can be trained to approximate nonlinear functions of the input because the neural population encoding is itself nonlinear. The reconstructed signal is obtained from a linear decoding of the delayed population features,

$$\hat{x}(t) = \mathbf{z}(t)\mathbf{w} \ . \tag{9}$$

where $\mathbf{w} \in \mathbb{R}^{NM\times 1}$ is the vector of decoding weights.

Suppose we sample the input signal at times $t_1, \ldots, t_P$, store the target values in the vector $\mathbf{x} \in \mathbb{R}^{P\times 1}$, and store the corresponding feature vectors in the rows of $\mathbf{Z} \in \mathbb{R}^{P\times NM}$. The optimal linear decoding weights are computed by ridge-regularized least squares,

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \|\mathbf{Z}\mathbf{w} - \mathbf{x}\|_2^2 + \lambda\|\mathbf{w}\|_2^2 \ . \tag{10}$$

We set $\lambda = 0.01\,\sigma_{\max}^2$, where $\sigma_{\max}$ is the largest singular value of $\mathbf{Z}$.

3.5 Interpretation as an LTI Readout

The delay-augmented decoder can be interpreted as a linear time-invariant (LTI) readout acting on the population's spike trains. This is because each decoder channel applies only three operations to the spike trains: a fixed time delay, convolution with a fixed synaptic filter h_{τ_s}, and a weighted sum across channels. Each of these operations is linear and time-invariant, and their composition is therefore also LTI. In particular,

$$\hat{x}(t) = \sum_{n=1}^{N} \sum_{m=0}^{M-1} w_{n,m}(h_{\tau_s} * S_n)(t - d_m), \tag{11}$$

which can be rewritten as

$$\hat{x}(t) = \sum_{n=1}^{N} (g_n * S_n)(t), \tag{12}$$

where

$$g_n(t) = \sum_{m=0}^{M-1} w_{n,m} h_{\tau_s}(t - d_m) \tag{13}$$

is the effective temporal filter associated with neuron n. Thus, instead of assigning each neuron a single instantaneous weight, the delayed decoder assigns each neuron a short temporal filter. This allows the readout to use recent spike history, not just current activity, which is helpful when useful information is distributed over time.

3.6 Computational Cost and Practical Choice of Delays

Adding delays introduces several practical costs compared to the instantaneous baseline.

First, delayed decoding requires buffering spikes so that they can be delivered after the prescribed offsets. In contrast, the instantaneous decoder needs no delay buffer. The required buffer length is determined by the maximum delay $d_{\max}$, so longer delays increase the amount of stored spike history.

Second, under matched dimensionality, the number of decoder inputs and hence the number of multiply-accumulate operations in the linear readout is the same in both settings. The additional implementation cost comes from maintaining the delay buffer and routing each spike to the corresponding delayed channels.

Third, the delayed decoder introduces latency: although the instantaneous tap preserves an immediate decoding path, the full delayed feature vector is not available until the longest delay has elapsed. Thus, larger delays provide a longer temporal window to the decoder, but also increase the effective response lag.

Finally, increasing the number of taps improves the temporal resolution available to the decoder, but under matched dimensionality it also reduces the number of neurons in the population and therefore the diversity of tuning curves. In this work, we use $M = 3$ taps as a simple trade-off between temporal coverage and population size.

4 Experiments

4.1 Overview

We compared baseline readouts (which use non-delayed activity alone) with delay-based readouts in LIF and ALIF populations under matched dimensionality across several one-dimensional decoding benchmarks. Each trial consisted of two independent 10-second simulations, one used for training the decoder and one for evaluating test performance.

All simulations were implemented in Nengo [2], which provides the spiking neuron dynamics, synaptic filtering, and connection infrastructure. The per-neuron delay mechanism was implemented as a custom Nengo node that buffers spikes and replays them at the appropriate offsets, as described in Sect. 3.3.

4.2 Setup

Benchmark Signals and Targets. We evaluated the decoder on a continuous one-dimensional reconstruction task in which the input signal $x(t)$ was band-limited white noise and the decoding target was the input signal itself. To probe how temporal scale affects decoding, we varied the cutoff frequency of the input across conditions (e.g. 1 Hz, 5 Hz, 10 Hz), producing signals with different characteristic timescales. Training and test signals were generated independently for each run and bandwidth condition, using different random seeds.

Network Parameters. All experiments used a synaptic filter time constant $\tau_s = 5\,\text{ms}$ and a simulation timestep $dt = 0.1\,\text{ms}$. For LIF neurons, we used the default Nengo parameters ($\tau_m = 20\,\text{ms}$, $\tau_{\text{ref}} = 2\,\text{ms}$). ALIF neurons shared the same base parameters with adaptation strength $\beta = 0.03$ and adaptation time constant $\tau_a = 1.5\,\text{s}$. In the delay-augmented condition, each neuron contributed $M = 3$ taps: an instantaneous channel at $d_0 = 0$ and two delayed channels at offsets d_1 and d_2. The nonzero delays were matched to the temporal scale of the input signal, with larger delays for slower signals and shorter delays for faster ones. In the primary experiments, the delayed population comprised $N_{\text{delay}} = 100$ neurons and the baseline used $N_{\text{inst}} = 300$ neurons, giving the same total number of decoder inputs in both conditions. We also repeated the comparison at larger proportional population sizes (for example, $N_{\text{delay}} = 300$ and $N_{\text{inst}} = 900$) and observed the same qualitative pattern.

Matched Dimensionality. To compare delay-based and instantaneous decoding fairly, we matched the total number of inputs to the linear decoder across conditions. In the instantaneous setting, each of the N_{inst} neurons contributes one decoder input, so the decoder has N_{inst} input features. In the delayed setting, each of the N_{delay} neurons contributes M features, one for each delay tap, so the decoder has MN_{delay} input features in total, where M is the number of delay taps per neuron. In all matched-dimensionality experiments, population sizes were chosen such that

$$N_{\text{inst}} = MN_{\text{delay}}. \tag{14}$$

For example, with $M = 3$, a delayed population of 100 neurons was compared with an instantaneous population of 300 neurons.

Effect of Neuron Model and Delay Scaling. The usefulness of delay taps depends on the temporal scale of the signal. For slower-changing input signals, larger delays produced better results, whereas higher-frequency inputs required shorter delays to remain informative.

Specifically, the delay taps are scaled to the signal's characteristic timescale

$$\theta = \frac{1}{2\pi f_c}, \tag{15}$$

where f_c is the bandwidth of the input. The two taps are then defined as $d_1 = \eta_1 \theta$ and $d_2 = \eta_2 \, \theta$, with $\eta_1 \approx 0.42$ and $\eta_2 \approx 0.71$. These coefficients were selected using Optuna [1], a Bayesian hyperparameter optimization framework, by minimizing reconstruction MSE on a held-out validation segment. The resulting values performed well across the conditions tested here, though they may not be globally optimal and could benefit from further tuning for different network configurations or input statistics.

Evaluation Metric. Decoder performance was evaluated using mean squared error (MSE) between the target signal $x(t)$ and the reconstructed output $\hat{x}(t)$. To compare delay-augmented decoding with the no-delay baseline, we report the relative percentage improvement in MSE:

$$\Delta\text{MSE}(\%) = 100 \cdot \frac{\text{MSE}_{\text{no-delay}} - \text{MSE}_{\text{delay}}}{\text{MSE}_{\text{no-delay}}}. \tag{16}$$

Here, $\text{MSE}_{\text{delay}}$ denotes the reconstruction error obtained with delayed decoding, and $\text{MSE}_{\text{no-delay}}$ denotes the error obtained with instantaneous decoding. Positive values therefore indicate that adding delays improves reconstruction accuracy, while negative values indicate that the no-delay decoder performs better.

4.3 Results

Each reported result is averaged over 10 different runs with different train/test seed pairs.

Figure 1 summarizes the relative MSE improvement of delay-augmented decoding over the no-delay baseline across white-noise bandwidths and firing-rate regimes. The strongest and most consistent gains occur for the 5 Hz condition; for LIF neurons, the improvement ranges from 11.7% to 33.3%, while for ALIF neurons it ranges from 4.1% to 39.3%. For the 10 Hz condition, the gains are smaller and concentrated in intermediate firing regimes, peaking at 18.7% for LIF and 20.3% for ALIF. The 1 Hz condition reveals the clearest difference between neuron models. ALIF remains strongly positive across all firing-rate ranges, with improvements from 7.4% to 39.2%, whereas LIF becomes negative at moderate to high firing rates, dropping to -10.8% in the highest range.

In additional experiments, we also decoded nonlinear target functions of the input, such as $|x|$ and $\sin(2\pi x)$. The qualitative pattern was again the same. Overall, these results show that delay taps are most beneficial when useful information is distributed over recent spike history, and that this effect can be enhanced by adaptive dynamics.

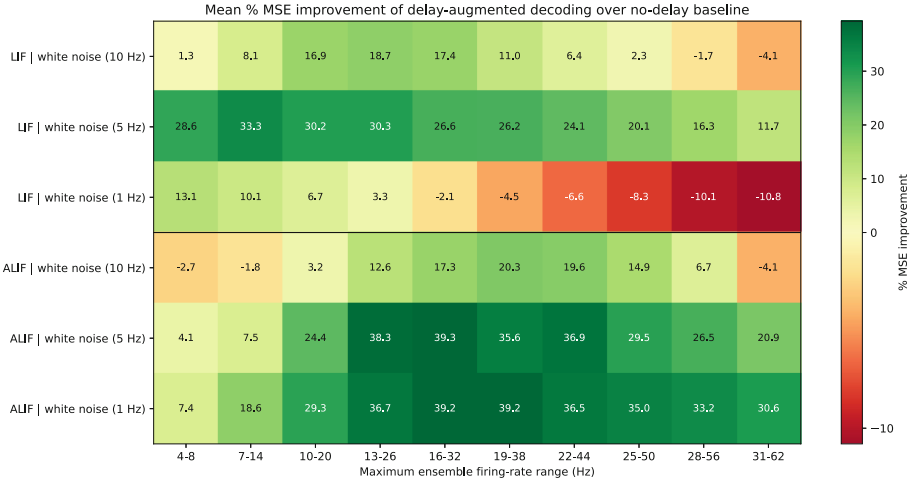

Fig. 1. Mean percentage MSE improvement of delay-augmented decoding over the no-delay baseline for LIF and ALIF populations under white-noise inputs with different frequencies. The no-delay population uses 300 neurons, while the delay-augmented population uses 100 neurons with $M = 3$ taps per neuron (one instantaneous, two delayed), yielding matched decoder dimensionality.

A complementary view is provided by the spike-count/error curves in Fig. 2, which plot test MSE against mean spikes per neuron per second. For LIF neurons, the delayed decoder consistently lies below the no-delay baseline across the plotted range, indicating lower reconstruction error despite using one third as many neurons. For ALIF neurons, the delayed decoder offers the clearest advantage at low and intermediate firing rates, where it reaches substantially lower MSE than the 300-neuron no-delay baseline, while the gap narrows at the highest firing rates. This pattern supports the interpretation that delay augmentation is most useful in regimes where spikes are informative but not yet abundant.

These trends support the interpretation in Sect. 3.5. Delay augmentation gives each neuron a short temporal filter instead of just one instantaneous weight. For ALIF neurons, whose responses depend on history because adaptation builds up from recent spikes, this temporal filter aligns better with the structure of the encoded signal. The weaker and more regime-dependent gains for LIF suggest that delayed readout features are most useful when the neural response depends strongly on recent history, as in ALIF populations with adaptation.

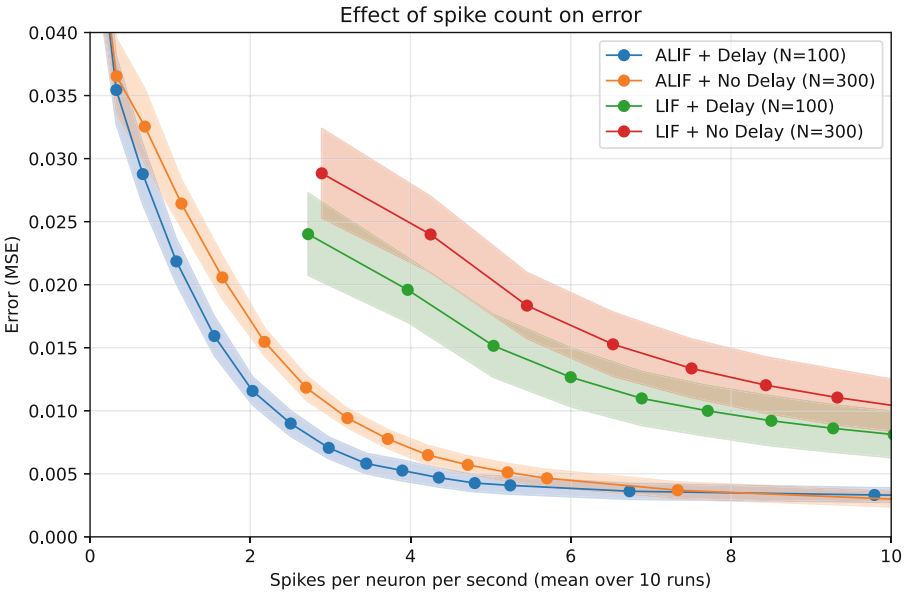

Fig. 2. Reconstruction error vs. spike rate for LIF and ALIF populations with and without delay augmentation under 5 Hz white-noise input, using matched decoder dimensionality. Here, N denotes the number of neurons in the population. Lines show the mean over 10 runs, and shaded bands indicate ± 1 standard deviation across runs.

5 Conclusion and Future Work

We studied a simple way to add temporal context to linear decoding in feedforward spiking networks: fixed per-neuron delay taps between the population and the readout. Under matched decoder dimensionality, delay augmentation frequently improved reconstruction accuracy, compared to an instantaneous decoder, and the largest gains appeared for ALIF populations, consistent with the fact that adaptation makes their responses depend on recent history. The results therefore suggest that even without recurrence, learned delays, nonlinear readouts, or increased population size, a small delay bank can substantially improve the accuracy–efficiency trade-off of spiking population decoding.

These findings are also consistent with the hypothesis that the brain may use adaptive neurons as a strategy to simultaneously maximize decoding accuracy and minimize energy expenditure, at least for rate-based readouts.

Several directions remain open. First, the present results suggest that delay spacing should be matched to the signal timescale, so a more systematic procedure for selecting the number and placement of delays would be useful. Second, future experiments should compare models under explicitly matched total spike budgets and hardware cost, since buffering delayed spikes introduces additional memory and routing overhead. Third, it would be valuable to evaluate the same

readout on richer temporal benchmarks, multidimensional signals, and downstream control tasks.

Acknowledgments. We gratefully acknowledge NSERC, Intel, and the University of Waterloo for financial and in-kind support of this work.

Disclosure of Interests. Authors declare no competing interests.

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: a next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2019)
2. Bekolay, T.: Nengo: a Python tool for building large-scale functional brain models. Front. Neuroinform. **7**, 48 (2014). https://doi.org/10.3389/fninf.2013.00048
3. Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., Maass, W.: Long short-term memory and learning-to-learn in networks of spiking neurons. In: Advances in Neural Information Processing Systems 31, Curran Associates, Inc (2018)
4. Davies, M., et al.: Loihi: a neuromorphic manycore processor with on-chip learning. IEEE Micro **38**(1), 82–99 (2018). https://doi.org/10.1109/MM.2018.112130359
5. Deckers, L., Van Damme, L., Van Leekwijck, W., Tsang, I.J., Latré, S.: Co-learning synaptic delays, weights and adaptation in spiking neural networks. Front. Neurosci. **18**, 1360300 (2024). https://doi.org/10.3389/fnins.2024.1360300
6. Eliasmith, C., Anderson, C.H.: Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems. MIT Press, Cambridge, MA (2003)
7. Gerwinn, S., Macke, J.H., Bethge, M.: Bayesian population decoding of spiking neurons. Front. Comput. Neurosci. **3**, 21 (2009). https://doi.org/10.3389/neuro.10.021.2009
8. Gerwinn, S., Macke, J.H., Bethge, M.: Reconstructing stimuli from the spike times of leaky integrate-and-fire neurons. Front. Neurosci. **5**, 1 (2011). https://doi.org/10.3389/fnins.2011.00001
9. Hammouamri, I., Khalfaoui-Hassani, I., Masquelier, T.: Learning delays in spiking neural networks using dilated convolutions with learnable spacings. In: The Twelfth International Conference on Learning Representations (2024)
10. Maass, W.: Networks of spiking neurons: the third generation of neural network models. Neural Netw. **10**(9), 1659–1671 (1997). https://doi.org/10.1016/S0893-6080(97)00011-7
11. Maass, W., Natschläger, T., Markram, H.: A model for real-time computation in generic neural microcircuits. In: Advances in Neural Information Processing Systems 15, MIT Press (2002)
12. Natschläger, T., Maass, W.: Information dynamics and emergent computation in recurrent circuits of spiking neurons. In: Advances in Neural Information Processing Systems 16, MIT Press (2003)
13. Naud, R., Gerstner, W.: Coding and decoding with adapting neurons: a population approach to the peri-stimulus time histogram. PLoS Comput. Biol. **8**(10), e1002711 (2012). https://doi.org/10.1371/journal.pcbi.1002711
14. Paninski, L., Pillow, J., Lewi, J.: Statistical models for neural encoding, decoding, and optimal stimulus design. In: Progress in Brain Research, vol. 165, pp. 493–507. Elsevier (2007). https://doi.org/10.1016/S0079-6123(06)65031-0

15. Pfeiffer, M., Pfeil, T.: Deep learning with spiking neurons: opportunities and challenges. Front. Neurosci. **12** (2018). https://doi.org/10.3389/fnins.2018.00774
16. Roy, K., Jaiswal, A., Panda, P.: Towards spike-based machine intelligence with neuromorphic computing. Nature **575**(7784), 607–617 (2019). https://doi.org/10.1038/s41586-019-1677-2
17. Salaj, D., Subramoney, A., Kraisnikovic, C., Bellec, G., Legenstein, R., Maass, W.: Spike frequency adaptation supports network computations on temporally dispersed information. eLife **10**, e65459 (2021). https://doi.org/10.7554/eLife.65459
18. Salinas, E., Abbott, L.F.: Vector reconstruction from firing rates. J. Comput. Neurosci. **1**(1–2), 89–107 (1994). https://doi.org/10.1007/BF00962720
19. Shrestha, S.B., Orchard, G.: SLAYER: Spike layer error reassignment in time. In: Advances in Neural Information Processing Systems, p. 31 (2018)
20. Sun, P., Chua, Y., Devos, P., Botteldooren, D.: Learnable axonal delay in spiking neural networks improves spoken word recognition. Front. Neurosci. **17**, 1275944 (2023). https://doi.org/10.3389/fnins.2023.1275944
21. Truccolo, W., Eden, U.T., Fellows, M.R., Donoghue, J.P., Brown, E.N.: A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. J. Neurophysiol. **93**(2), 1074–1089 (2005). https://doi.org/10.1152/jn.00697.2004
22. Voelker, A.R., Eliasmith, C.: Improving spiking dynamical networks: accurate delays, higher-order synapses, and time cells. Neural Computation **30**(3), 569–609 (2018). https://doi.org/10.1162/neco_a_01046
23. Voelker, A.R., Kajic, I., Eliasmith, C.: Legendre memory units: continuous-time representation in recurrent neural networks. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
24. Wang, X., Lin, X., Dang, X.: A delay learning algorithm based on spike train kernels for spiking neurons. Front. Neurosci. **13**, 252 (2019). https://doi.org/10.3389/fnins.2019.00252
25. Westover, M.B., Eliasmith, C., Anderson, C.H.: Linearly decodable functions from neural population codes. Neurocomputing **44–46**, 691–696 (2002). https://doi.org/10.1016/S0925-2312(02)00459-9
26. Wu, W., Gao, Y., Bienenstock, E., Donoghue, J.P., Black, M.J.: Bayesian population decoding of motor cortical activity using a Kalman filter. Neural Comput. **18**(1), 80–118 (2006). https://doi.org/10.1162/089976606774841585
27. Zhang, M., et al.: Supervised learning in spiking neural networks with synaptic delay-weight plasticity. Neurocomputing **409** (2020). https://doi.org/10.1016/j.neucom.2020.03.079