# Three-Factor Delay Learning Rules Spiking Neural Networks

Luke Vassallo and Nima Taherinejad
*Institute for Computer Engineering*
*Heidelberg University*
Heidelberg, Germany
{luke,nima}@ziti.uni-heidelberg.de

***Abstract*—Spiking Neural Networks (SNNs) are dynamical systems that operate on spatiotemporal data, yet their learnable parameters are often limited to synaptic weights, contributing little to temporal pattern recognition. Learnable parameters that delay spike times can improve classification performance in temporal tasks, but existing methods rely on large networks and offline learning, making them unsuitable for real-time operation in resource-constrained environments. In this paper, we introduce synaptic and axonal delays to leaky integrate and fire (LIF)-based feedforward and recurrent SNNs, and propose three-factor learning rules to simultaneously learn delay parameters online. We employ a smooth Gaussian surrogate to approximate spike derivatives exclusively for the eligibility trace calculation, and together with a top-down error signal determine parameter updates. Our experiments show that incorporating delays improves accuracy by up to 20% over a weights-only baseline, and for networks with similar parameter counts, jointly learning weights and delays yields up to 14% higher accuracy. On the SHD speech recognition dataset, our method achieves similar accuracy to offline backpropagation-based approaches. Compared to state-of-the-art methods, it reduces model size by 6.6× and inference latency by 67%, with only a 2.4% drop in classification accuracy. Our findings benefit the design of power and area-constrained neuromorphic processors by enabling on-device learning and lowering memory requirements.**

***Index Terms*—SNN, Delay Learning, LIF, Online Learning**

## I. Introduction

Spiking Neural Networks (SNNs) are continuous time dynamical systems that integrate a weighted sum of action potentials and emit a spike when sufficiently stimulated. Unlike Artificial Neural Networks (ANNs), which perform static data transformations without an explicit temporal component, SNNs inherently operate in the time domain. However, despite this fundamental difference, SNNs still rely primarily on synaptic weights for learning [1], [2]. The absence of dedicated mechanisms for capturing temporal dynamics often leads to lower task performance, as shown by state-of-the-art methods for learning temporal delays [3]–[5]. Alternatively, achieving competitive performance often requires significantly larger models with orders of magnitude more parameters [1]. Recently improved techniques for training SNNs such as surrogate gradient methods [6] [7], have allowed learning of time-dependent parameters on a large scale such as synaptic delay learning.

Learnable delays add a degree of freedom that facilitates temporal pattern detection. For instance, in Figure 1 the green plots shows the effect two incident spikes have on the membrane potential. The first spike at $t_1$ increases the membrane potential

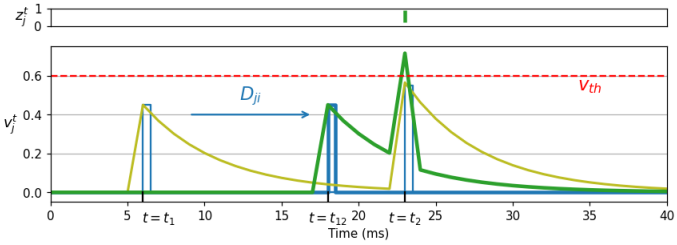


Fig. 1: Illustration of coinciding pre-synaptic spikes leading to post-synaptic. Two distinct pre-synaptic neurons emit spikes at $t = t_1$ and $t = t_2$, respectively. Introducing a delay to the first spike shifts its effect to $t = t_{12}$, where it coincides with the second spike and thereby triggering downstream effects.

in proportion to the strength of the synaptic weight and decays towards zero. A second spike at a later time $t = t_2$ has a similar effect, but since the two do not coincide in time, the membrane potential does not exceed the threshold and thus, the neuron does not fire. However, if $t_1$ is delayed to $t_{12}$ by $D_{ji}$, then the spikes co-incide and, as the bold trace shows, a spikes it emitted. The converse can also be useful, increasing the separation of temporally local spikes can suppress firing.

Our work introduces three-factor delay learning rules for updating synaptic and axonal delays in a gradient-equivalent manner. We explore the application of these rules to keyword spotting (KWS) tasks using feedforward SNNs and spiking recurrent neural networks (SRNNs). Our results highlight the contribution of each delay type to classification accuracy and demonstrate the advantages of heterogeneous spatiotemporal parameters over homogeneous spatial ones. Our proposed method, along with its systematic evaluation, provides insights for chip designers balancing model versatility and efficiency in size weight and power (SWaP)-constrained neuromorphic processors.

## II. Background

Delays in point neurons can be categorized based on morphology. At the synapse, delays arise from the complex chemical reactions triggered by the arrival of an action potential at the pre-synaptic site, which later initiates the post-synaptic response [8]. If the postsynaptic response causes the neuron to fire, the resulting spike propagates along the axon introducing an axonal

delay [9]. Before reaching the synapse of a downstream neuron, the signal must also traverse the dendritic arbor, leading to a dendritic delay that depends on dendritic morphology, membrane properties, and the presence of active ion channels [10].

### A. Delay learning

Several approaches exist to embed and learn delays in neural networks. Spike layer error reassignment in time (SLAYER) [11] addresses the temporal credit assignment problem in SNNs, enabling learning both synaptic weights and delays in multi-layer networks. Sun et al. [12] introduces axonal delays by extending the underlying spike response model (SRM) with temporal convolutions through rectified axonal delay (RAD) or the neurophysiologically inspired variable axonal delay (VAD) [5]. In both cases, the delay kernel gradient was calculated numerically using the finite difference approximation and demonstrated an accuracy improvement of up to 18.7% on KWS datasets [7]. Deckers et al. [3] further adapts SLAYER's [11] axonal delay mechanism to individual synapses and when combined with a constrained adaptive leaky integrate and fire (ADLIF) neuron model [1], this approach achieves accuracy comparable to temporal convolution methods [4].

However since in principle the adaptive neuron model and synaptic delays both facilitate temporal pattern recognition, it is unclear how much each feature contributes to classification accuracy. An alternative approach, implemented in PyTorch uses dilated convolution with learnable spacings (DCLS) [4], [13]. It employs a Gaussian kernel parameterized by weight and delay parameters, that is convolved with the incident spike train, to generate the spatiotemporal response of a synapse.

By leveraging automatic differentiation, it achieves state-of-the-art accuracy even on challenging KWS datasets. Recognizing the benefits of this method in sparse settings, Mészáros et al. [14] introduces dynamic pruning to enforce sparsity.

Event propagation [15] provides an alternative training algorithm that uses spike times to compute exact gradients without surrogates. Applied to keyword spotting and related tasks, event propagation has achieved state-of-the-art performance [16], with subsequent experiments highlighting the importance of loss shaping [17].

### B. Online learning

Backpropagation is update-locked [18], preventing parameter updates until the full input sequence is processed. For sequence data, this results in linearly increasing memory demands, which restricts real-time and on-chip implementation. Online learning releases this constraint and traces its origins to Real-Time Recurrent Learning (RTRL) [19]. By accumulating the gradients of the hidden states, RTRL eliminated the need to unroll the network over some time window $T$. However, its cubic memory complexity ($\mathcal{O}(n^3)$) is prohibitive compared to backpropagation through time (BPTT) ($\mathcal{O}(nT)$). Subsequent research has focused on approximating RTRL to reduce computational costs [20], [21] or developing biologically plausible three-factor learning rules [22] that aim to retain BPTT's performance guarantees.

Three-factor learning rules mitigate the biological implausibility of BPTT-like algorithms [15], [23] by addressing the weight transport [24] and update locking [18] problems. They rely on eligibility traces, composed of pre- and post-synaptic factors that capture temporal activity. However, unlike two-factor learning rules [25], synaptic changes occur only when these traces are modulated by a third factor, for instance, a top-down learning signal spatially propagating task-relevant error information (see [26] for a comprehensive survey). Superspike [27] constructs an eligibility trace by combining a low-pass filtered trace of pre-synaptic activity with a nonlinear function of post-synaptic activity, while the error signal serves as the third factor. By contrast, Eligibility Propagation (e-prop) [28] reformulates BPTT as the three-factor learning rule framework, demonstrating equivalence for feedforward networks and approximating it for recurrent architectures. Dynamics for Deep Continuous Local Learning (DECOLLE) [29] computes local synthetic gradients using per-layer readout functions. [30] decouples spatial gradients (top-down learning signals within the same timestep) and temporal gradients (eligibility traces spanning timesteps). It is equivalent to BPTT for shallow SRNNs and approximates its performance for deeper architectures.

## III. Proposed Methods

We present the neural dynamics for point neurons, network architecture and three-factor delay learning rules using e-prop [28] as a basis for online learning.

### A. Neuron dynamics

The leaky integrate and fire (LIF) model stands out for its computational efficiency and biological plausibility, and has become the defacto standard model in neuromorphic engineering [31], [32]. For a point neuron $j$ the LIF model integrates the weighted sum of input spikes into a one-dimensional hidden variable $v_j^t$, representing the membrane potential. When the membrane potential exceeds a pre-defined threshold, $v_{thr}$ the neuron generates an action potential or spike, $z_j^t \in \{0, 1\}$, which propagates along its axon.

The model consists of two components: a first-order linear differential equation describing the evolution of the membrane potential and a spike generation mechanism. The leaky integration of synaptic currents into the membrane potential $v_j(t)$ is given by

$$\tau_m \frac{dv_j}{dt} = -(v_j(t) - v_{reset}) + RI_j(t) \tag{1}$$

where $\tau_m$ is the membrane time constant, $v_{reset}$ is the membrane potential at rest, $R$ is the input resistance, and $I_i(t)$ is the input synaptic current at time, $t$. The synaptic current, described by Equation (2) is stateless and represented as a weighted sum between the synaptic weight, $W_{ji}^{in}$ and the delayed pre-synaptic input $x_i(t - D_{ji}^{in})$. The synaptic $D_{ji}^{in} \in \mathbb{Z}$, or $D_i^{in} \in \mathbb{Z}$ axonal delay is described in number of timesteps, bounded by $D_{max}$.

$$I_j(t) = \sum_i W_{ji}^{in} x_i(t - D_{ji}^{in}) \tag{2}$$

Equation (1) is solved analytically under the assumptions $v_{\text{reset}} = 0$ and $R = (1 - e^{-\frac{\Delta t}{\tau_m}})^{-1}$. The solution is discretized

with a timestep $\Delta t$, yielding the discrete timestepped neuron dynamics in Equation (3):

$$v_j^t = \alpha v_j^{t-1} + \sum_j W_{ji}^{\text{in}} x_i^{t-D_{ji}^{in}} \tag{3}$$

where $\alpha = e^{-\frac{\Delta t}{\tau_m}}$ is the decay factor determined by the membrane time constant $\tau_m$ and the timestep $\Delta t$.

The spike reset mechanism extends the membrane potential dynamics with a thresholding operation, mathematically represented using the Heaviside function:

$$z_j^{t+1} = H(v_j^t > v_{thr}) \tag{4}$$

where $z_j^{t+1}$ is a binary variable indicating the presence of a spike at time $t+1$. Following the spike event, the membrane potential is explicitly reset by deducting $v_{thr}$. The full dynamics for an SNN are thus given by:

$$v_j^{t+1} = \alpha v_j^t + W_{ji}^{in} x_i^{t-D_{ji}^{in}} - z_j^t v_{th} \tag{5}$$

The neuron dynamics can be extended with recurrent connections yielding an SRNN with optional delays.

$$v_j^{t+1} = \alpha v_j^t + W_{ji}^{rec} z_j^{t-D_{ji}^{rec}-1} + W_{ji}^{in} x_i^{t-D_{ji}^{in}} - z_j^t v_{th} \tag{6}$$

*1) Surrogate gradient:* The spiking mechanism described by Equation (4) is discontinuous and therefore non-differentiable. It blocks the flow of gradient information and is effectively eliminated by means of a surrogate function used only during the calculation of parameter updates. We adopt the piecewise linear function in Equation (7) with $\gamma_{pd} = 0.3$.

$$\frac{dz}{dv_j^t} \approx \frac{\gamma_{pd}}{v_{th}} \max\left(0, 1 - \left|\frac{v_j - v_{th}}{v_{th}}\right|\right) \tag{7}$$

### B. Neural network architecture and loss function

The neural network topology consists of a set of virtual neuron, followed a hidden layer without optional recurrent connections, and an output readout layer. Virtual input neurons apply the spiking input signal with an optional axonal delay. The single hidden layer neurons characterized by weights and optional synaptic delays, calculates the neural dynamics as described in Equation (5) for SNNs and Equation (6) for SRNNs.

The LIF neuron output connects to a readout layer comprised of a set of leaky integrate (LI) neurons whose dynamics are also described by Equation (3), however, they omit the spike reset mechanism. The leakage for the readout is defined with by $\kappa = e^{-\frac{\Delta t}{\tau_o}}$, where $\tau_o$ is the membrane time constant.

A large membrane time constant ($\tau_{out} \geq 1s$) is assigned to the LI neurons in the readout layer, allowing it to practically hold the state indefinitely.

For k-class classification, we assume $k$ categories represented as a K-dimensional one-hot encoded vector. A softmax function is applied to the output from the readout layer, and the cross-entropy loss is computed $E = -\sum_k \sum_t \pi_k^t log(\hat{\pi}_k^t)$, where $\pi_k^t$ represents the ground-truth classification label for class $k$ and $\hat{\pi}_k^t$ denotes the corresponding probability predicted by the network.

### C. Online learnable delays

Eligibility Propagation (e-prop) [28] is an online learning algorithm that closely approximates BPTT while describing synaptic weight updates as three-factor learning rules. The formulation supports delay-parametrized spike trains but does not permit the delays to be learnable. This work proposes the three-factor learning rules in Equation 8 using the same refactorisation for calculating error gradients with respect to network parameters describing temporal delays. $\frac{dE}{dD_{ji}}$ represents the error gradients with respect to synaptic delays where, $D_{ji} \in \{D_{ji}^{in}, D_{ji}^{rec}\}$. Similarly, $\frac{dE}{dD_i}$ corresponds to the gradient with respect to axonal delays, where $D_i \in \{D_i^{in}, D_j^{rec}\}$.

$$\frac{dE}{dD_{ji}} = \sum_t \frac{dE}{dz_j^t} \cdot \left[\frac{dz_j^t}{dD_{ji}}\right]_{\text{local}}, \tag{8}$$

The top down learning signal $L_j^t = \frac{dE}{dz_j^t}$ is derived from the cross-entropy loss function applied to the readout layer outputs defined in Section III-B.

The eligibility trace $e_{ji}^t = \frac{\partial z_j^t}{\partial v_j^{t'}}$ incorporates information about the previous spiking activity and can be recursively expressed, permitting real-time implementation as shown in Equation (9). Here, $\frac{\partial z_j^t}{\partial v_j^{t'}}$, is the surrogate gradient of the neuron's output with respect to the hidden state calculated with Equation (7), $\frac{\partial v_j^t}{\partial v_j^{t-1}} = \alpha$ is computed by differentiating Equation (5) with respect to $v_j^t$, and $\epsilon_{ji}^{t-1}$ is the eligibility vector.

$$e_{ji}^t = \frac{\partial z_j^t}{\partial v_j^{t'}} \left(\frac{\partial v_j^t}{\partial v_j^{t-1}} \cdot \epsilon_{ji}^{t-1} + \frac{\partial v_j^t}{\partial D_{ji}}\right) \tag{9}$$

Calculating the eligibility trace entails computing the derivative of the hidden state described by Equation (6) with respect to the delay parameters, $\frac{dv_j^t}{dD_{ji}}$. However since $D_{ji}$ is associated with spike trains (input $x_i^{t-D_{ji}^{in}}$ or recurrent $z_j^{t-D_{ji}^{rec}}$) the derivative does not exist. We overcome this problem by representing the spike with a continuous function, [4], [33], [34], particularly the Gaussian kernel in Equation (11).

### D. Spike train kernel

A Gaussian kernel was chosen because preliminary experiments showed that causal kernels, such as the exponential, did not support effective learning, whereas both anti-causal and symmetric kernels did. Between the Gaussian and the double exponential kernels, the Gaussian consistently performed better, likely due to its smoother profile. The non-causal nature of the surrogate spike derivative is undesirable for real-time applications and introduces an implementation cost requiring a ring buffer (see Section Section VI).

Equation (10) describes a spike train parametrized with synaptic delay $D_{ji}$ as a sum of $k$ dirac delta functions delayed in time by $t_k + D_{ji}$,

$$x_i^{t-D_{ji}} = \sum_k \delta(t - t_k - D_{ji}) \tag{10}$$

where $\delta$ is the Dirac delta function, $t_k$ are the spike times, and $D_{ji}$ is the synaptic delay between the pre-synaptic neuron $j$ and the post-synaptic neuron $i$. The Dirac delta function is approximated with a Gaussian and parameterized by the synaptic delay, $D_{ji}$ ($D_i$ for axonal delay), where $D_{ji} = -\frac{D_{max}-1}{2}$ represents no delay and $D_{ji} = \frac{D_{max}-1}{2}$ represents the maximum delay, similar to [4]. Equation (11) describes the delay parametrized Gaussian kernel. During parameter updates, $D_{ji}$ is clamped within $\pm\frac{D_{max}-1}{2}$.

$$x_i^{t-D_{ji}} \approx \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t-t_k-D_{ji})^2}{2\sigma^2}} \tag{11}$$

The derivative of Equation (11) with respect to the delay parameters can now be calculated, yielding:

$$\frac{dx_i^{t-D_{ji}}}{dt} = -\frac{t-t_k-D_{ji}}{\sqrt{2\pi}\sigma^3} e^{-\frac{(t-t_k-D_{ji})^2}{2\sigma^2}} \tag{12}$$

## IV. Experiments

Using KWS tasks [7], we systematically evaluate the proposed three-factor delay learning rules against offline methods [4] using dense and sparse SNNs. The experiments extend to SRNNs, with ablation tests to identify where delays impact task performance. Hammouamri et. al's DCLS-based method [4] was selected as a baseline because it uses a LIF model and achieves state-of-the-art performance using temporal convolutions together with BPTT. Synaptic delays are introduced at input synapses for SNNs and additionally at recurrent synapses for SRNNs. Axonal delays are applied to virtual input neurons for SNNs and to both input and recurrent neurons for SRNNs.

### A. Dataset

We use the spiking heidelberg digits (SHD) and spiking speech commands (SSC) datasets to evaluate our implementation of online delay learning. Both datasets consist of spoken sequences in spiking format, generated by preprocessing the original audio using an artificial cochlear model, which converts the auditory signals into spikes [7]. The SHD dataset consists of 10,000 professionally recorded samples from 12 participants and contains spoken sequences of the digits zero through nine in both English and German. SSC dataset is derived from the speech commands (SC) dataset [35] and contains 100,000 samples of 35 commonly spoken keywords collected from a diverse group of participants under varying recording conditions.

The datasets consist of 700 spiking neurons with a temporal resolution of 1 ms. To reduce computational time and GPU memory requirements, the spatial dimension is binned by a factor of six, resulting in $700/6 = 116$ input neurons. Additionally, the temporal dimension is sub-sampled at a rate of 10ms yielding sequences of approximate 100 samples in length.

### B. Experimental Setup

We evaluate online delay learning in SNNs and SRNNs across three configurations: a large fully connected network with 128 hidden neurons, an 80% sparse variant, and a small fully connected network with 16 hidden units. Sparsity is implemented with fixed random binary maps. Two weights-only configurations, featuring 32 and 64 hidden units, are included to compare performance based on parameter counts. SNNs experiments are complemented by evaluating the same configurations using BPTT-based DCLS, and for this, we extend the state-of-the-art [4] with axonal delays. The online learning experiments are repeated using SRNNs with both input and recurrent delays (DCLS experiments are omitted because offline implementation is unsupported for SRNNs). Evaluations on the SSC dataset follow an identical strategy and are limited to 256-neuron fully connected configurations.

Learnable delays tend to offer a smaller contribution compared to weights [14]. To assess the efficacy of delay learning, we perform control experiments. For SNNs, we evaluate delay learning as a function of sparsity under fixed and co-learned conditions. For SRNNs, we fix sparsity and analyze the effect of placing delays at the input, recurrent layer, or both.

The learning rates for weights and delays are set to $10^{-4}$ and $10^{-2}$, respectively. Training is conducted with a batch size of 16 samples over 60 epochs for both the SHD dataset and SSC datasets. Synaptic and axonal delays are limited to 25 timesteps thereby introducing a maximum latency of 250ms. The inference pass uses binary spike trains and the Gaussian kernel is only invoked during parameter updates. Offline DCLS-based experiments retain all original hyperparameters [4]. We train all networks on the training dataset and evaluate top-1 classification accuracy on the test set. Each experiment is repeated five times, and test accuracy is reported with a 95% confidence interval, assuming a t-distribution. The implementation is performed in PyTorch 2.1 and runs on a Debian 12 system with a Xeon 6526Y CPU, 1 TB of memory, and a 20 GB RTX A4500 Ampere GPU.

## V. Results

Table I presents the top-1 test classification accuracy obtained by SNN and SRNN configurations described in Section IV-B on the SHD and SSC datasets. Comparing weights-only SNNs with online learning to the DCLS-based offline implementation reveals a 15% improvement in our favor. This large gap was unexpected, but it can likely be explained by our precise analytical LIF dynamics, rather than the direct Euler integration used in the SpikingJelly-based [36] baseline. Recent studies show that this type of discretization can pose challenges in stability and parameterization [37]. For configurations with synaptic and axonal delays, our implementation achieves test accuracies for fully connected models that closely match backpropagation within 0.15%. This result falls within statistical bounds, demonstrating that our approach is equivalent to offline methods under the same model configurations. In sparse and small network configurations, the inclusion of delays improves performance, surpassing BPTT by 1% and 4.3% respectively.

Temporal delays bring a significant improvement in test classification accuracy. In fully connected networks, accuracy increases by 13.2% with the addition of synaptic delays and by a slightly lower 12.6% with axonal delays. The gap widens in sparse and small models, with accuracy gains reaching 18% and 16.8% for synaptic and axonal delays, respectively. Under sparse SNN conditions, learnable delays contribute 3% and 2.3% in classification accuracy for synaptic and axonal delays, respectively, compared to fixed random delays. For

TABLE I: Test Classification accuracy for SNN and SRNN models trained with online and offline (BP) methods.

| ID | Dataset | Configuration | Weights only | | Fixed synaptic delays, learnable weights | Learnable weights, and synaptic delays | | Fixed axonal delays, learnable weights | Learnable weights, and axonal delays | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Accuracy (%) | Params (k) | Accuracy (%) | Accuracy (%) | Params (k) | Accuracy (%) | Accuracy (%) | Params (k) |
| 1 | | FC[†]-128 SNN (BP[§]) | 63.70% ± 1.6% | 17.4 | 90.19% ± 0.27% | 92.79% ± 1.1% | 32.3 | 74.20% ± 1.07% | 92.02% ± 1.63% | 17.5 |
| 2 | | SP[‡]-128 SNN (BP) | 59.72% ± 3.9% | 3.5 | 84.27% ± 3.32% | 88.42% ± 2.7% | 6.5 | 48.36% ± 3.66% | 86.84% ± 2.84% | 3.5 |
| 3 | | FC-16 SNN (BP) | 50.33% ± 6.8% | 2.2 | 76.07% ± 3.79% | 82.79% ± 1.0% | 4.0 | 56.57% ± 2.24% | 78.96% ± 6.31% | 2.3 |
| 4 | SHD | FC-128 SNN | 79.46% ± 0.2% | 17.4 | 92.42% ± 0.22% | 92.64% ± 0.4% | 32.3 | 91.90% ± 0.4% | 92.01% ± 0.3% | 17.5 |
| 5 | | SP-128 SNN | 71.23% ± 1.4% | 3.5 | 86.26% ± 0.79% | 89.22% ± 0.8% | 6.5 | 86.36% ± 0.6% | 88.06% ± 2.2% | 3.5 |
| 6 | | FC-16 SNN | 69.09% ± 1.4% | 2.2 | 83.87% ± 2.47% | 85.58% ± 4.3% | 4.0 | 84.93% ± 1.4% | 84.92% ± 0.7% | 2.3 |
| 7 | | FC-32 SNN | 73.88% ± 1.52% | 4.35 | - | - | - | - | - | - |
| 8 | | FC-64 SNN | 77.69% ± 2.14% | 8.75 | - | - | - | - | - | - |
| 9 | | FC-128 SRNN | 85.77% ± 0.94% | 33.8 | 91.73% ± 0.81% | 92.36% ± 0.3% | 65.0 | 91.66% ± 0.6% | 91.64% ± 1.6% | 34.0 |
| 10 | | SP-128 SRNN | 83.52% ± 2.38% | 6.76 | 90.18% ± 0.20% | 91.40% ± 0.71% | 13.00 | 89.27% ± 1.40% | 90.42% ± 0.39% | 6.81 |
| 11 | | FC-16 SRNN | 73.29% ± 1.02% | 2.43 | 83.92% ± 5.80% | 85.81% ± 2.88% | 4.54 | 84.09% ± 4.32% | 83.49% ± 6.58% | 2.56 |
| 12 | SSC | FC-256 SNN (BP) | 43.16% ± 0.16% | 34.82 | 69.67% ± 0.10% | 72.87% ± 0.03% | 64.51 | 43.12% ± 2.00% | 67.62% ± 0.30% | 34.93 |
| 13 | | FC-256 SNN | 52.24% ± 0.05% | 34.82 | 70.93% ± 0.24% | 71.90% ± 0.06% | 64.51 | 69.35% ± 0.48% | 70.18% ± 0.04% | 34.93 |
| 14 | | FC-256 SRNN | 68.86% ± 0.05% | 100.35 | 73.52% ± 0.01% | 74.66% ± 0.09% | 195.58 | 72.37% ± 0.14% | 73.18% ± 0.07% | 100.72 |

[†] FC = Fully Connected, [‡] SP = Sparse (80%), [§] BP = Backpropagation

small networks, the amount is smaller for synaptic delays and negligible for axonal delays.

Simultaneous weight and delay learning improves the model's accuracy-to-parameter ratio. Sparse models with synaptic delays (6.5k parameters) achieve 11.5% higher classification accuracy than 64-wide fully connected model with 8.75k parameters. Sparse models with axonal delays (3.5k parameters) provide an even greater improvement of 14.2% over a 32-wide fully connected model with 4.35k parameters. For smaller models, synaptic delays contribute an 11.7% accuracy increase, whereas axonal delays offer a higher improvement of 15.84%. Axonal delays bring the highest increase in test classification accuracy per added model parameter. However, the finer granularity provided by synaptic delays consistently results in higher accuracy, particularly in the presence of sparsity.

Three factor learning rules also facilitates training SRNN. Concerning the SHD dataset, weights-only SRNNs establish a baseline at 85.72% in fully connected configurations, 6.3% higher than SNNs. When adding input and recurrent delays, we observe comparable performance with SNNs, albeit lagging behind by approximately 0.5% while showing increased robustness in sparse configurations. The classification accuracy on the SHD dataset seems to saturate at approximately 92.5% for our setup, despite SRNNs doubling the number of parameters and adding recurrent connections.

To investigate, Figure 3d presents ablation experiments examining the effect of input and recurrent delays on classification accuracy. Adding synaptic input delays or recurrent delays offers a 6.56% and 5.2% increase in accuracy respectively, while adding both offers 6.6%. Axonal delays follow a similar pattern with 0.5% lower accuracy on average. In sparse configurations, the inclusion of input and recurrent synaptic delays provides a 1.3% improvement in classification accuracy compared to input delays alone. Under these conditions, the accuracy loss remains at 0.95%, compared to a 3.4% loss in SNNs, despite an 80% reduction in parameters. Axonal delays exhibit a similar trend, albeit with smaller improvements.

The SSC dataset poses a more challenging task for keyword spotting evidenced by a larger performance gap between models with fixed and learnable delays. In contrast, for the SHD dataset, fully connected models—whether feedforward or recurrent—exhibited negligible differences in accuracy. The classification accuracy of SNNs remains comparable to offline implementations. The versatility of SRNNs with synaptic and axonal delays leads up to 2.8% and 3% higher classification accuracy, respectively, compared to SNNs.

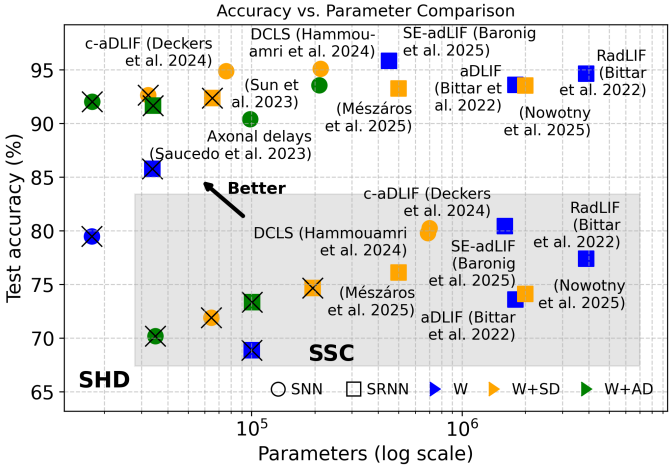


Fig. 2: Comparison against offline methods on SHD and SSC (gray background). Marker shape distinguishes model type, while color determines the type and presence of delays. Markers overlaid with an X correspond to the proposed online method

## VI. Comparison and Discussion

Figure 2 compares our online method (marked with X) against competing state-of-the-art approaches on the SHD and SSC datasets, where the latter are highlighted with a gray background. All competing methods are offline because they use BPTT or event-prop [15] to train their models, making them unsuitable for implementation on resource-constrained devices. Offline methods obtain highest accuracy with two [3], [4], [37] or even three hidden layers [1], while our method being approximate and constrained to a single layer achieves competitive performance within 3% of the best offline results. Relative to the DCLS baseline, our approach achieves a 2.5% lower test classification accuracy on SHD dataset. However, the loss of accuracy is offset by a 6.6× model size reduction, and since we only have

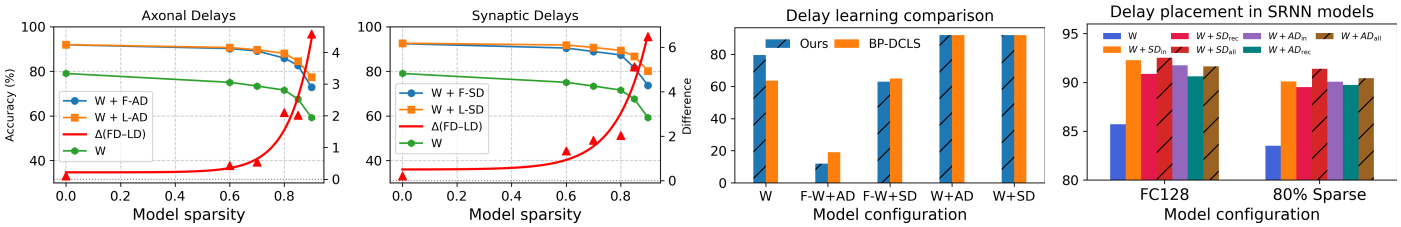


Fig. 3: Control experiments illustrating the efficacy of delay-learning. (a, b) Sweep model sparsity for axonal (AD) and synaptic delays (SD), with fixed (W+F*) or learnable (W+L*) delay parameters. (c) Compares learning delays with fixed weights (F-W*) using the proposed method and BPTT. (d) Compares SRNNs with delays in the input, recurrent, and all connections.

one delay layer (three in the baseline), our model also achieves 67% lower inference latency. Other competing solutions obtain marginally higher accuracy on the SHD albeit using a different model such as an adaptive SRNN [1], [3], [37] or event-based state space models [38], [39].

Similarly for the SSC dataset, however, being more challenging we find the more versatile SRNN offers 3% higher accuracy, and therefore an advantage over feed-forward networks. However, having both more classes and varying recording conditions, our methods lags behind by 5.6% in classification accuracy. We anticipate that incorporating multiple layers could further enhance accuracy on both datasets, but would be particularly beneficial for SSC.

### A. *Efficacy of delay learning*

The major contribution to test classification accuracy is offered by the addition of fixed delays. As shown in Table Table I, co-learning weights and delays in dense situations offers only a modest 1-2% accuracy improvement, as a result we conducted further experiments to further understand the imapct of delay learning. The efficacy of delay learning becomes more apparent under highly sparse conditions, where contributions up to 5-6% over fixed delays can be observed in Figures 3a and 3b. The three-factor delay learning rules approximate backpropagation because the online recursive calculation of parameter updates, which enables their real-time implementation, relies entirely on historical information. The comparison of the local learning rules to backpropagation in Figure 3c shows that the approximation holds for synaptic parameters with a gap of 2%, while for axonal parameters it starts to break down as the gap widens to 8%. The widened gap for axonal delays might be explained by the approximate parameter updates amassing into large errors, since each axonal parameter at the input neuron is derived from the accumulated contributions of all postsynaptic neurons. The significant drop in classification accuracy for F-W+AD in Figure 3c is likely due to the small number of trainable parameters, totaling 116. However, as learning under sparse conditions shows, when co-learned the impact of these errors could potentially have a regularization effect and overall yield acceptable results. These inaccuracies could potentially be mitigated by adopting alternative coding strategies, such as time-to-first-spike (TTFS) coding, which would allow axonal delays to be assigned locally to the hidden neuron rather than the input neuron.

### B. *Relevance for on-chip implementation*

Three factor delay learning rules offer a solution to the update locking problem [18] enabling real-time on-device gradient calculations that closely approximate backpropagation. The resources of such systems are dominated by on-chip Static Random Access Memory (SRAM) [40], [41] which accommodates both model parameters and the implementation of neural dynamics. Axonal delays scale linearly while synaptic delays scale quadratically with layer size which impacts the memory requirements for both parameter storage as well as their implementation. Beyond FIFO buffers for delaying input spikes, or ring buffers for delayed synaptic integration, the real-time calculation of delay parameter updates requires implementing the product between the eligibility trace and the surrogate kernel at the time the spike reaches the post-synaptic neuron. When no assumption can be made on the spike quantity for any given timestep the implementation relies on ring buffers. Their size is smallest when the input spike is delayed requiring a depth equal to the kernel length, but when pre-computing updates can reach a maximum equal to the larger between the kernel length and the maximum delay. Alternatively when activations are sparse, the post synaptic potentials can be buffered while keeping track of the spike times. The added versatility can significantly reduce the size of the model while improving accuracy. Considering a typical quantisation of the setup in Section IV-B, with 8-bit weights, 5-bit delays, and 16-bit membrane potentials, the implementation of learnable axonal delays increases memory usage by 8.3% of which 6.2% are needed for learning. For synaptic delays this increases to 57.7%, of which a third are needed for learning. Using a dense weights-only model as the baseline, a model that incorporates axonal or synaptic delays must reach at least 43% and 74% sparsity, respectively, in order to achieve the same memory footprint when both parameter storage and implementation overhead are considered.

## VII. Conclusion

In this work, we proposed three-factor learning rules for synaptic and axonal delays in SNNs and SRNNs, validated their implementation against state-of-the-art BPTT-based learning methods by demonstrating comparable performance, and studied

the efficacy of delay learning through ablation experiments. Under online conditions, we demonstrated that delay learning improves classification accuracy by up to 20% in SNNs and 12.5% in SRNNs and is particularly advantageous for small networks, where co-learning weights and delays yield up to 14% accuracy improvement over a weights-only model of equivalent size. While further research is required to scale to cognitively demanding tasks, online delay learning offers a practical solution to enhance embedded systems with limited on-chip SRAM with adaptability without compromising accuracy, and enables trade-offs between implementation complexity and task performance.

## Author Contributions

L.V. proposed the learning rule, associated experiments and their implementation in PyTorch, performed the literature review and wrote the initial version of the manuscript. N.T. provided supervision and funding. All authors discussed the results, reviewed, and edited the manuscript.

## Funding

This work was supported by the Hector Stiftung (grant no. 2304191) and the Field of Focus 2 at Heidelberg University (grant no. 33477).

## References

[1] A. Bittar and P. N. Garner. A surrogate gradient spiking baseline for speech command recognition. *Frontiers in Neuroscience*, 16:865897, August 2022.

[2] B. Yin *et al.* Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. *Nature Machine Intelligence*, 3(10):905–913, October 2021.

[3] L. Deckers *et al.* Co-learning synaptic delays, weights and adaptation in spiking neural networks. *Frontiers in Neuroscience*, 18:1360300, April 2024.

[4] I. Hammouamri *et al.* Learning delays in spiking neural networks using dilated convolutions with learnable spacings. In *The Twelfth International Conference on Learning Representations*, 2024.

[5] P. Sun *et al.* Learnable axonal delay in spiking neural networks improves spoken word recognition. *Frontiers in Neuroscience*, 17:1275944, November 2023.

[6] E. O. Neftci *et al.* Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks. *IEEE Signal Processing Magazine*, 36(6):51–63, November 2019.

[7] B. Cramer *et al.* The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(7):2744–2757, July 2022.

[8] B. L. Sabatini and W. G. Regehr. Timing of neurotransmission at fast synapses in the mammalian brain. *Nature*, 384(6605):170–172, November 1996.

[9] D. Debanne *et al.* Axon Physiology. *Physiological Reviews*, 91(2):555–602, April 2011.

[10] M. London and M. Häusser. DENDRITIC COMPUTATION. *Annual Review of Neuroscience*, 28(1):503–532, July 2005.

[11] S. B. Shrestha and G. Orchard. Slayer: spike layer error reassignment in time. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 1419–1428, Red Hook, NY, USA, 2018. Curran Associates Inc.

[12] P. Sun *et al.* Axonal Delay as a Short-Term Memory for Feed Forward Deep Spiking Neural Networks. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8932–8936, Singapore, Singapore, May 2022. IEEE.

[13] I. Khalfaoui-Hassani *et al.* Dilated convolution with learnable spacings. In *The Eleventh International Conference on Learning Representations*, 2023.

[14] B. Mészáros *et al.* Learning Delays Through Gradients and Structure: Emergence of Spatiotemporal Patterns in Spiking Neural Networks, November 2024. arXiv:2407.18917 [cs].

[15] T. C. Wunderlich and C. Pehle. Event-based backpropagation can compute exact gradients for spiking neural networks. *Scientific Reports*, 11(1):12829, June 2021.

[16] B. Mészáros *et al.* Efficient Event-based Delay Learning in Spiking Neural Networks, June 2025. arXiv:2501.07331 [cs].

[17] T. Nowotny *et al.* Loss shaping enhances exact gradient learning with Eventprop in spiking neural networks. *Neuromorphic Computing and Engineering*, 5(1):014001, March 2025.

[18] M. Jaderberg *et al.* Decoupled neural interfaces using synthetic gradients. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1627–1635. PMLR, 06–11 Aug 2017.

[19] R. J. Williams and D. Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280, June 1989.

[20] F. Benzing *et al.* Optimal Kronecker-sum approximation of real time recurrent learning. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 604–613. PMLR, 09–15 Jun 2019.

[21] C. Tallec and Y. Ollivier. Unbiased online recurrent optimization. In *International Conference on Learning Representations*, 2018.

[22] W. Gerstner *et al.* Eligibility Traces and Plasticity on Behavioral Time Scales: Experimental Support of NeoHebbian Three-Factor Learning Rules. *Frontiers in Neural Circuits*, 12:53, July 2018.

[23] D. E. Rumelhart *et al.* Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[24] Q. Liao *et al.* How important is weight symmetry in backpropagation? In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 1837–1844. AAAI Press, 2016.

[25] G.-q. Bi and M.-m. Poo. Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type. *The Journal of Neuroscience*, 18(24):10464–10472, December 1998.

[26] O. Marschall *et al.* A unified framework of online learning algorithms for training recurrent neural networks. *J. Mach. Learn. Res.*, 21(1), January 2020.

[27] F. Zenke and S. Ganguli. SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks. *Neural Computation*, 30(6):1514–1541, June 2018.

[28] G. Bellec *et al.* A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, 11(1):3625, July 2020.

[29] J. Kaiser *et al.* Synaptic Plasticity Dynamics for Deep Continuous Local Learning (DECOLLE). *Frontiers in Neuroscience*, 14:424, May 2020.

[30] T. Bohnstingl *et al.* Online Spatio-Temporal Learning in Deep Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2022.

[31] L. Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Research Bulletin*, 50(5-6):303–304, November 1999.

[32] W. Gerstner *et al. Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 1 edition, July 2014.

[33] J. Göltz *et al.* DelGrad: Exact event-based gradients in spiking networks for training delays and weights, December 2024. arXiv:2404.19165 [cs].

[34] X. Wang *et al.* A Delay Learning Algorithm Based on Spike Train Kernels for Spiking Neurons. *Frontiers in Neuroscience*, 13:252, March 2019.

[35] P. Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *CoRR*, abs/1804.03209, 2018.

[36] W. Fang *et al.* Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480, 2023.

[37] M. Baronig *et al.* Advancing spatio-temporal processing through adaptation in spiking neural networks. *Nature Communications*, 16(1):5776, July 2025.

[38] M. Schöne *et al.* Scalable Event-by-Event Processing of Neuromorphic Sensory Signals with Deep State-Space Models. In *2024 International Conference on Neuromorphic Systems (ICONS)*, pp. 124–131, Arlington, VA, USA, July 2024. IEEE.

[39] T. Soydan *et al.* S7: Selective and Simplified State Space Layers for Sequence Modeling, October 2024. arXiv:2410.03464 [cs].

[40] C. Frenkel and G. Indiveri. ReckOn: A 28nm Sub-mm2 Task-Agnostic Spiking Recurrent Neural Network Processor Enabling On-Chip Learning over Second-Long Timescales. In *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, pp. 1–3, San Francisco, CA, USA, February 2022. IEEE.

[41] M. Davies *et al.* Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro*, 38(1):82–99, January 2018.