TU Delft

Delft University of Technology

Document Version
Final published version

Licence
Dutch Copyright Act (Article 25fa)

Citation (APA)
Cojocaru, A., Micheli, A., Booij, O., & Tömen, N. (2026). Precise Spike Timing Meets Discrete Time: Discretization Effects in Spiking Neural Networks. In I. Koprinska, J. Mendes-Moreira, & P. Branco (Eds.), *Machine Learning and Principles and Practice of Knowledge Discovery in Databases - International Workshops of ECML PKDD 2025, Revised Selected Papers* (pp. 380-394). (Communications in Computer and Information Science; Vol. 2840 CCIS). Springer. https://doi.org/10.1007/978-3-032-19099-4_27

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.



Precise Spike Timing Meets Discrete Time: Discretization Effects in Spiking Neural Networks

Alexandru Cojocaru, Aurora Micheli(✉), Olaf Booij, and Nergis Tömen

Delft University of Technology, Delft, The Netherlands
a.micheli@tudelft.nl

Abstract. Spiking neural networks (SNNs) are inspired by the continuous-time dynamics of biological neurons, yet their practical implementations typically rely on discrete-time approximations to accommodate digital hardware and time-stepped neuromorphic systems. This discretization, while necessary, introduces numerical errors that may affect training dynamics and model performance, especially in systems where precise spike timing encodes critical information. Recent work has proposed gradient-based training methods that treat spike times as differentiable continuous variables, enabling temporally precise learning. However, the impact of discretizing these spike times during deployment remains poorly understood. In this paper, we systematically investigate the effects of spike time discretization using a representative multi-spike model as a case study. We propose a causal discretization scheme and analyze its quantization error both theoretically and empirically. Our results show that this error depends not only on the time step size Δt but also on the temporal structure of the input. We demonstrate that certain structured inputs can produce local minima in discretization error, and we explore whether such minima correspond to optimal choices of Δt for real datasets. Experiments on MNIST, Fashion MNIST, and EMNIST reveal that discretization can influence both classification accuracy and network sparsity. Finally, we compare training and evaluation under mismatched and aligned time regimes, examining how consistency in spike time discretization affects performance and sparsity in spike-timing-based SNNs.

Keywords: Spiking Neural Networks · Spike time discretization · Classification tasks

1 Introduction

Spiking Neural Networks (SNNs) are a class of artificial neural networks that more closely mimic the operational principles of biological brains, transmitting

A. Cojocaru and A. Micheli—These authors contributed equally to this work and share first authorship.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
I. Koprinska et al. (Eds.): ECML PKDD 2025, CCIS 2840, pp. 380–394, 2026.
https://doi.org/10.1007/978-3-032-19099-4_27

information through discrete, temporally sparse events known as spikes [1,14,20]. This event-driven computation enables SNNs to operate with high efficiency and low energy consumption, especially when implemented on neuromorphic hardware optimized for sparse and asynchronous processing [2,7,15,21,27].

A common approach to encode information via spikes in SNNs is rate coding, where the average firing rate of a neuron over time is used to represent input features or signal intensity. This approach offers a robust and practical mechanism for input encoding and align closely with how computations are performed in conventional (non-spiking) Artificial Neural Networks (ANNs). It has also enabled SNNs to achieve competitive performance on standard machine learning and computer vision benchmarks such as MNIST [18], FashionMNIST [26], and CIFAR-10 [17]. Therefore, most SNN models rely primarily on rate-based representations, where the precise timing of individual spikes is of secondary importance and spike times are often randomized using stochastic encodings like Poisson spike trains [8,24]. Yet, this approach does not fully exploit the sparsity and energy efficiency that spiking neural networks can offer, and it often relies on generating a large number of spikes.

Also, despite the continuous-time formulation underlying most SNN models, where membrane potentials and synaptic currents evolve according to differential equations, practical implementations typically rely on numerical discretization schemes, such as Euler integration [11,22]. This choice ensures compatibility with conventional digital computation and time-stepped neuromorphic systems, which operate on fixed clock cycles [7,15]. Consequently, most SNN models deployed on neuromorphic platforms are discretized in time, even though they are inspired by the analog dynamics of biological systems.

At the same time, there is growing evidence from neuroscience and machine learning that precise spike timing can serve as a powerful computational resource [12,16,25]. In many real-world applications, such as processing event-based sensor data [19], modeling temporal patterns, and performing time-series analysis, timing is a core component of the signal. Leveraging the full expressiveness of time as a continuous variable offers the potential for significantly more efficient and temporally accurate information processing. Recent models such as [3,28] and [13], building on early work like SpikeProp [5], explore this regime by deriving and deploying closed-form expressions for precise spike times and computing gradients with respect to these continuous variables. These approaches treat time itself as a differentiable quantity, allowing the network to harness the structure embedded in temporal dynamics. However, while such models are defined and operate in continuous time, the hardware they typically run on, whether conventional CPUs/GPUs or digital neuromorphic chips, remains discrete [7,15,23]. This inevitably introduces numerical errors and discrepancies between the theoretical model and its practical implementation. To the best of our knowledge, there have been no studies investigating how this spike time discretization, and the resulting discretization errors, affect the model's behavior and performance.

For this reason, in this work we examine the effects of discretizing precise spike times, using as a case study the model proposed in [3], which extends the framework introduced in [13] by allowing multiple spikes per neuron. We make the following contributions and address the corresponding research questions:

- We introduce a causal spike time discretization scheme and handle its non-differentiability to enable effective training.
- We analytically characterize the discretization error introduced by mapping continuous spike times to discrete time bins, showing that it scales with the time step size Δt, but also depends on the temporal structure of the input. Using synthetic patterns, we show that structured spiking inputs, such as bursts, can lead to local minima in the quantization error. We then investigate whether similar effects are observable in real-world datasets.
- Building on our analysis of the discretization error, we examine how varying time bin sizes affect network training across three different datasets, focusing on key metrics such as accuracy and network sparsity.
- We investigate how aligning the spike time representations (discrete or continuous) between training and inference affects the network performance and behavior.

2 Methods

We start by introducing the spiking neuron model in Sect. 2.1, followed by our approach to incorporating time discretization into this model in Sect. 2.2.

2.1 The Neuron Model

Consistently with the formulation in [3], we consider a neural network composed of CuBa (Current-Based) LIF (Leaky Integrate-and-Fire) neurons with a soft reset of the membrane potential. The membrane potential represents the neuron's internal state, which evolves over time in response to incoming signals. Its dynamics are governed by a system of linear ordinary differential equations:

$$\frac{du^{(l,j)}}{dt} = -\frac{1}{\tau} u^{(l,j)}(t) + g^{(l,j)}(t) - \vartheta \delta\left(u^{(l,j)}(t) - \vartheta\right) \tag{1}$$

$$\frac{dg^{(l,j)}}{dt} = -\frac{1}{\tau_s} g^{(l,j)}(t) + \sum_{i=1}^{N^{(l-1)}} w_{i,j}^{(l)} \sum_{z=1}^{n^{(l-1,i)}} \delta\left(t - t_z^{(l-1,i)}\right) \tag{2}$$

$$\delta(x) = \begin{cases} +\infty & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Here, $u^{(l,j)}$ denotes the membrane potential of the j-th neuron in layer l. The term $n^{(l,j)}$ indicates the number of spikes emitted by the j-th neuron in layer

l, and N^l is the number of neurons in layer l. The variable $g^{(l,j)}$ represents the post-synaptic current resulting from incoming spikes. Each spike received from a pre-synaptic neuron increases $g^{(l,j)}$ by an amount equal to the corresponding synaptic weight $w_{i,j}^{(l)}$. The function $\delta(x)$ denotes the Dirac delta function, which satisfies $\int_{-\infty}^{+\infty} \delta(x), dx = 1$. We refer to pre-synaptic and post-synaptic connections as those between adjacent layers. The parameters τ and τ_s are the membrane and synaptic time constants, respectively, governing the exponential decay of the membrane potential and post-synaptic current over time. When the membrane potential $u^{(l,j)}$ reaches the threshold ϑ, the neuron emits a post-synaptic spike at time $t_k^{(l,j)}$, where k denotes the index of the emitted spike. Immediately following this event, $u^{(l,j)}$ is reset to zero by an instantaneous negative current equal in magnitude to the threshold ϑ.

As shown in [13] and [3], the CuBa LIF neuron can be mapped to a more general formulation known as the Spike Response Model (SRM) [9]. By assuming that the membrane time constant is twice the synaptic time constant ($\tau = 2\tau_s$), this mapping allows for the analytical derivation of the exact spike time $t_k^{(l,j)}$. This eliminates the need for perpetual numerical integration, enabling event-driven computation, which means that computations only happen when a spike is generated from a neuron.

2.2 Spike Time Discretization

As discussed in Sect. 2.1, the model described in [3] admits a closed-form solution for spike timing, enabling precise computation of spike events. However, when deployed on clock-based hardware such as GPUs, CPUs, or digital neuromorphic chips, time must be discretized. This enforced discretization can introduce numerical errors and potential instabilities, leading to discrepancies between the theoretical model and its practical implementation.

To investigate the impact of these effects, rather than relying on numerical integration, we introduce a novel time discretization scheme using fixed time steps Δt during both inference and backpropagation. Spike times, originally computed in continuous time, are then aligned to these discrete steps. Specifically, each spike is delayed to the nearest subsequent multiple of Δt, preserving causality and the model's core dynamics. Let $t_k^{(l,j)}$ denote the k-th spike time of neuron j in layer l. Its discretized counterpart, $\tilde{t}_k^{(l,j)}$, satisfies the alignment condition:

$$0 \equiv \tilde{t}_k^{(l,j)} \pmod{\Delta t} \tag{4}$$

Because spikes in discrete time cannot occur before their continuous counterparts, each spike is delayed to the nearest future multiple of Δt:

$$\tilde{t}_k^{(l,j)} = t_k^{(l,j)} + \Delta t - \mathrm{mod}(t_k^{(l,j)}, \Delta t) \tag{5}$$

We define the discretization function as:

$$d(t_k^{(l,j)}) = t_k^{(l,j)} + \Delta t - \mathrm{mod}(t_k^{(l,j)}, \Delta t) = \tilde{t}_k^{(l,j)} \tag{6}$$

This function is non-differentiable with respect to $t_k^{(l,j)}$, implying that gradients from the original model cannot be directly reused. Different strategies have been proposed to handle the non-differentiability of activation functions during backpropagation both for ANNs and SNNs, including surrogate gradients and biased estimators [4,22,28,29]. For our experiments, which use a single hidden layer, we adopt the straight-through estimator (STE) [4], a simple yet effective biased gradient estimator that performs well in shallow networks. The STE is used to approximate the non-differentiable spike time discretization function (Eq. 6) as an identity during backpropagation:

$$\frac{\partial \tilde{t}_k^{(l,j)}}{\partial t_k^{(l,j)}} \approx 1 \tag{7}$$

Thus, the chain rule simplifies to:

$$\frac{\partial \mathcal{L}}{\partial t_k^{(l,j)}} \approx \frac{\partial \mathcal{L}}{\partial \tilde{t}_k^{(l,j)}}$$

In practice, we apply the estimator as follows: the forward pass uses discretized spike times to compute outputs and propagate post-synaptic effects, while the backward pass uses the original continuous spike times to compute gradients. This allows us to retain the benefits of discrete-time execution without modifying the gradient flow.

3 Experiments

The following sections expand on the points outlined in Sect. 1. In Sect. 3.1, we define the discretization error and examine its dependence on the temporal structure of the input, using both synthetic patterns and real-world datasets. Section 3.2 analyzes how different time bin sizes affect classification accuracy and network sparsity. Finally, Sect. 3.3 explores whether using a consistent spike time representation, either discrete or continuous, during both training and testing has a beneficial impact.

3.1 Quantization Error in Spike Time Discretization

As explained in Sect. 2.2, in order to preserve causality, our approach maps the precise time stamp of each spike to the discretized time stamp corresponding to the endpoint of each time bin. While we call it time discretization, we assume a setting in which precise spike times encode information within the network. Thus, from a signal processing perspective, the deviation between the original continuous-valued time stamps and their discretized counterparts can be interpreted as quantization error. In deep networks, this quantization error might compound to disrupt the representation and transmission of information between layers.

Under typical conditions, quantization error depends linearly on the quantization step size, which, in our case, is the time bin Δt. For instance, assume that the quantization error, E_Q, is defined as the root mean square error (RMSE) between the original and quantized time stamps. In the typical scenario of a uniform quantizer, where the original time stamps are mapped to the middle point of each time bin, it is easy to show [10] that the quantization RMSE $E_Q^{\text{RMSE}} = \Delta t/\sqrt{12}$. For our case of causally mapping each time stamp to the endpoint of the time bin, we can write the quantization RMSE as

$$E_Q^{\text{RMSE}} = \sqrt{\frac{1}{\Delta t}\int_0^{\Delta t} [g(t)]^2 dt} = \frac{\Delta t}{\sqrt{3}}, \tag{8}$$

where

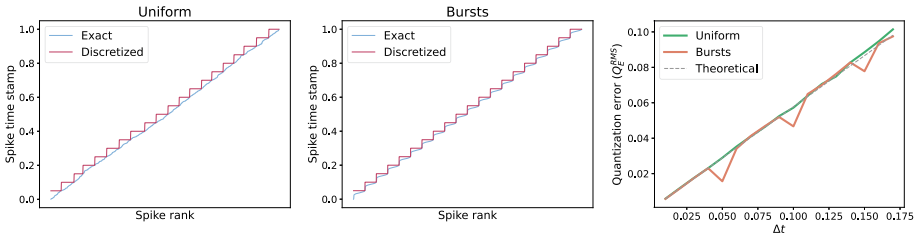
$$g(t) = -t + \Delta t \tag{9}$$

is the deviation of the discretized time stamp within one time bin. However, this calculation assumes a uniform distribution of time stamps within each time bin. When this assumption is violated, for example, when spikes occur in 'bursts' with periodic intervals, it is possible to find time bins $\Delta t > 0$ which locally minimize the quantization RMSE. As a demonstration, we consider a stochastic ramp input. Figure 1a shows spike times drawn from a uniform distribution in the interval $[0, 1]$ (blue) and their discretized values (red). For discretization, we use $\Delta t = 0.05$. In contrast, if we generate spikes with periodic bursts, it is possible to find Δt which agree well with the characteristic time scale of the bursts (Fig. 1b). Figure 1c shows the quantization RMSE for both cases as a function of Δt. As expected, the numerically computed quantization RMSE Q_E^{RMSE} of the uniformly drawn spikes (green) follows the theoretical value given in Eq. 8 (dashed gray line), while the bursts lead to local minima of the quantization RMSE (orange).

This comparison between the uniformly distributed and 'burst'-like spike trains shows that with a data-driven approach to choosing Δt, it might be possible to achieve a smaller quantization error than what would be theoretically expected with a prior uniformity assumption. Here, our underlying assumption is that real-world data will have some periodicity with characteristic time scales. Therefore, we hypothesize that it might be possible to use a larger Δt, reducing computational load and improving sparsity, while achieving a similar quantization error, and potentially a similar task accuracy, compared to using a smaller Δt.

We then extend this analysis to real-world datasets by computing the mismatch between precise and discretized spike times at the input layer for three image datasets: MNIST [18], Fashion MNIST (FMNIST) [26] and Extended MNIST (EMNIST) [6]. Pixel intensities are converted to spike times using latency coding, where brighter pixels (higher intensities) produce earlier spikes. Each normalized pixel intensity I_p (relative to the maximum intensity $I_{\max}$) is scaled by the simulation window $T_{\max}$:

$$t_{\text{spike}} = T_{\max}\left(1 - \frac{I_p}{I_{\max}}\right)$$



(a) (b) (c)

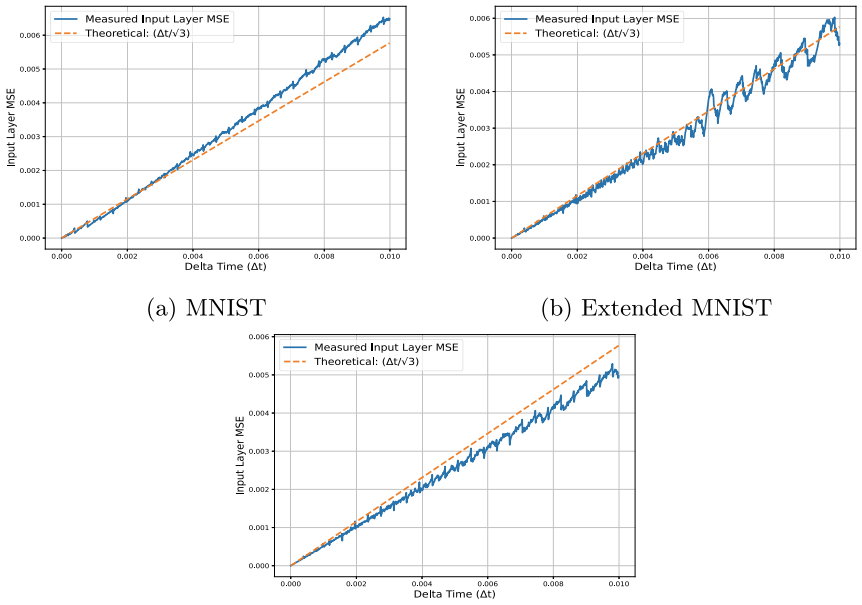
Fig. 1. Quantization RMSE for spike times generated using two different mechanisms: **(a)** sampling from a uniform distribution, and **(b)** periodic "burst" patterns characterized by a non-stationary firing rate with a specific time scale. Sampled exact spike times are shown in blue and the corresponding discretized value is shown in red. **(c)** When sampling is uniform, quantization RMSE Q_E^{RMSE} decreases linearly (green) with quantization step Δt, following Eq. 8 (dashed gray line). However, when spikes are emitted in periodic bursts, quantization RMSE displays local minima (orange) when Δt matches the burst frequency well. This means a smart, data-driven choice of Δt can yield a lower Q_E^{RMSE} than would be expected from a ramp function. (Color figure online)

For the real-world datasets, we report the mean squared error (MSE), instead of the RMSE, to improve visibility. For each sample in the dataset, we compute the MSE between original and discretized spike times and average over the dataset. We then repeat this for different time bin sizes Δt in the range $[0.0, 0.01]$. Figure 2 shows both the theoretical MSE (orange), derived from Eq. 8, and the empirical MSE (blue). For MNIST, the MSE increases roughly linearly with the time step size. However, for Fashion MNIST, and even more prominently for Extended MNIST, the MSE exhibits clear local minima and maxima, suggesting that the underlying structure of spike patterns can interact with the discretization scale to reduce quantization mismatch at specific time bin sizes. This phenomenon becomes noticeable only at sufficiently large values of Δt.

3.2 Training Under Varying Time Bin Sizes

In this section, we investigate how varying the time bin size affects network training. As discussed previously, while the model in [3] is defined and operates in continuous time, its deployment on conventional hardware requires discretization in fixed time steps. The size of these time bins can vary and plays a critical role in bridging the gap between theoretical models and their practical implementations.

In Sect. 3.1, we showed that smaller time bins Δt result in a smaller quantization error under the assumption that spikes are uniformly distributed within each bin. Yet, this assumption often does not hold in practice, as real datasets may exhibit non-uniform temporal structures such as burstiness or periodicity. In such cases, it is possible to find non-zero time bin sizes $\Delta t > 0$ that locally minimize the quantization error (Fig. 2b). Here, we extend that analysis to investigate whether selecting a time bin size that locally minimizes quantization error, when allowed by the dataset's structure, also leads to improved training performance.



(a) MNIST (b) Extended MNIST

(c) Fashion MNIST

Fig. 2. Comparison of theoretical MSE (orange) and empirical MSE (blue) between precise and discretized spike times at the input layer for MNIST (a), Extended MNIST (b), and Fashion MNIST (c). The theoretical MSE is computed analytically based on spike time quantization, while the empirical MSE is measured by comparing actual spike trains before and after discretization. Across datasets, MNIST shows a roughly linear increase in MSE with time step size, whereas Fashion MNIST and especially Extended MNIST exhibit local minima and maxima, suggesting that the temporal structure of spike patterns might interact with discretization to reduce quantization error at specific time bin sizes. (Color figure online)

For our experiments, we again considered three different datasets: MNIST [18], Fashion MNIST (FMNIST) [26] and Extended MNIST (EMNIST) [6]. The MNIST and FMNIST datasets each consist of 60,000 training samples and 10,000 test samples, where each sample is a 28×28 pixel grayscale image representing one of ten classes. EMNIST is a larger and more diverse extension of MNIST, encompassing both digits and letters across 47 classes.

Table 1. Hyperparameter configurations used for training on MNIST, EMNIST, and Fashion MNIST.

Hyperparameter	MNIST	EMNIST	Fashion MNIST
Layer Configuration	784-100-10	784-200-47	784-400-400-10
Training Batch Size	50	50	50
Test Batch Size	100	100	100
Learning Rate	0.003	0.003	0.0005
Optimizer	Adam	Adam	Adam
Target Spike Count (True Class)	15	15	15
Target Spike Count (False Class)	3	3	3
τ_s (Hidden & Output Layers)	0.13	0.13	0.13
Threshold (Output Layer)	0.2	1.3	0.7
Threshold (Hidden Layer)	1.3	0.2	0.13
Threshold (Hidden Layer 2)	-	-	0.45
Weight Distribution	U(-1, 1)	U(-1, 1)	U(-1, 1)
Epochs	30	30	30
Number of Experiments	3	3	3
Simulation Time Window	0.1 s	0.1 s	0.1 s

The network and hyperparameter settings largely follow the original implementation in [3], with minor modifications to the architecture. In particular, we use smaller layer sizes for MNIST and EMNIST to reduce the risk of overfitting. Table 1 summarizes the network architectures and hyperparameter settings used in our experiments across the three datasets. We train the networks for 30 epochs using different values of the time bin size Δt and compare the results to the original continuous-time model ($\Delta t = 0$). The Δt values were selected based on the MSE analysis in Sect. 3.1, specifically by identifying local minima and maxima of the MSE curve where such patterns were visible, depending on the dataset. Our goal is to investigate whether choosing theoretically optimal or suboptimal Δt values affects classification accuracy and model convergence. For completeness, we also extend the analysis to a broader range of Δt values. The loss function is computed as:

$$\mathcal{L} := \frac{1}{2} \sum_{j=1}^{N^{(o)}} \left(y - n^{(o,j)} \right)^2 \tag{10}$$

where y is the spike count target, set to 15 for the true class and 3 for all false classes across all datasets, and $n^{(o,j)}$ denotes the spike count of output neuron j in the output layer.

Each experiment was repeated three times, and we report the mean and variance of the results. Test accuracy and total spike count in the hidden layer

are shown as functions of training epochs for the three datasets: MNIST (Fig. 3), Extended MNIST (Fig. 4), and Fashion MNIST (Fig. 5).

While test accuracy provides a direct measure of the network's performance, spike count serves as an intrinsic metric reflecting the network's sparsity and potential power efficiency, as fewer spikes imply fewer computations and, consequently, lower energy consumption. Since multiple spikes from the same neuron occurring at exactly the same point in time are neither feasible nor biologically realistic, our discretisation procedure allows only one spike per neuron per time step. When multiple spikes fall within the same time bin, only the first is retained while the others are discarded. As a result, larger time bin sizes tend to reduce the total spike count. This observation is potentially important from an application perspective: different choices of time bin size may yield similar classification accuracy, yet lead to different levels of spiking activity. In scenarios where energy efficiency or sparsity is desirable, a more silent network may be preferable. This insight can help guide the selection of an appropriate discretisation scheme, particularly when the time bin size is a tunable parameter.

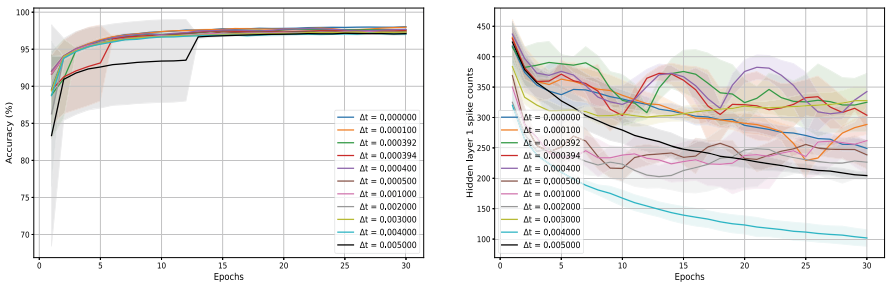


Fig. 3. Test accuracy (*left*) and spike count (*right*) on MNIST from epoch 0 to epoch 30. Accuracy remains largely stable for small to moderate Δt, while spike count decreases for larger Δt. Some time bin sizes (e.g. $\Delta t = 0.004$) achieve lower spike activity without compromising performance.

The results on MNIST (Fig. 3) and Fashion MNIST (Fig. 5) indicate that the network is generally robust to variations in the time bin size Δt. Only for relatively large time bins (e.g. when $\Delta t \gtrsim 0.004$ for Fashion MNIST) do we observe a noticeable decline in test accuracy. In contrast, smaller values of Δt lead to negligible differences in accuracy, regardless of their associated MSE values shown in Fig. 2. As expected, larger time bin sizes result in lower spike count at initialization. However, this trend does not necessarily persist throughout training, as the network may compensate by increasing activity. Notably, certain values of Δt, specifically, $\Delta t = 0.004$ for MNIST and $\Delta t = 0.00263$ for Fashion MNIST, consistently result in lower spike counts while maintaining comparable test accuracy. These settings may therefore be preferable, offering a favorable trade-off between performance and efficiency.

In the case of Extended MNIST, despite the presence of clear local minima in the MSE analysis (Fig. 2), the most pronounced occurring at $\Delta t = [0.00584, 0.00768]$, the corresponding test accuracy results suggest that selecting these Δt values does not necessarily lead to improved network performance. In fact, test accuracy appears to decrease consistently as the time step size increases, indicating that lower MSE does not always correlate with better classification outcomes. It is worth noting, however, that the Δt values selected based on the MSE analysis are likely too large and introduce excessive error, as also suggested by the significant reduction in spike count. We conclude that, for this dataset, smaller values of Δt not only yield higher accuracy but also lead to a more silent network with reduced spiking activity.

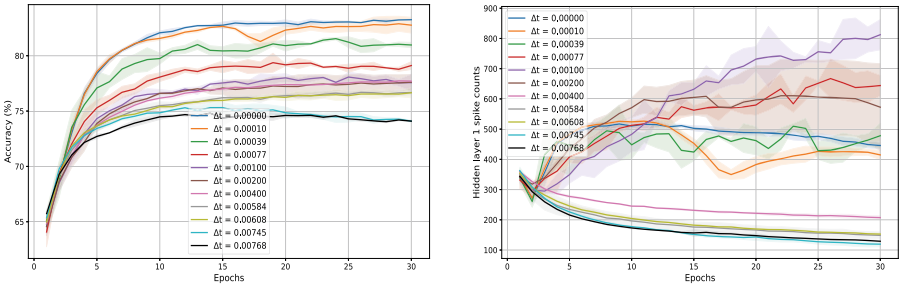


Fig. 4. EMNIST: Test accuracy (*left*) and spike count (*right*) from epoch 0 to epoch 30. Accuracy declines as the time bin size increases.

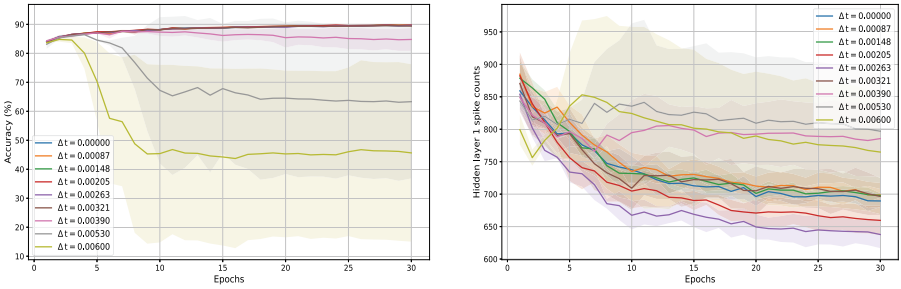
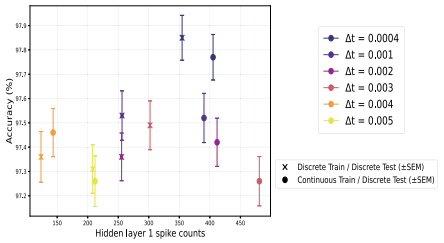


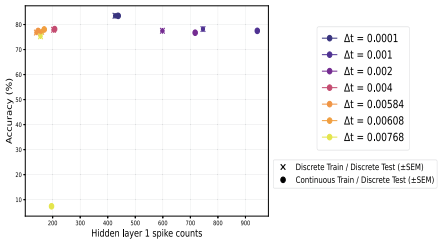
Fig. 5. Fashion MNIST: Test accuracy (*left*) and spike count (*right*) from epoch 0 to epoch 30. Accuracy remains stable for small Δt, with only larger time bins causing a minor drop. Specific Δt choices (e.g. $\Delta t = 0.00263$) reduce spike count while maintaining performance, offering a trade-off between sparsity and accuracy.

3.3 Training–testing Consistency in Spike Time Representation

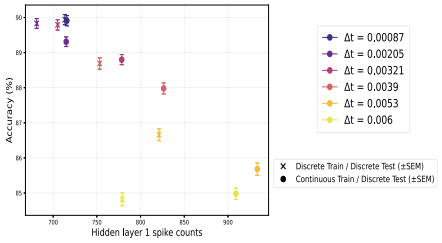
Lastly, we explore whether aligning the spike time representation, discrete or continuous, across both training and testing phases positively influences performance and sparsity. As in Sect. 3.2, we use the same three datasets: MNIST, Fashion MNIST, and Extended MNIST. Using identical hyperparameters and training settings, we evaluate the network on each dataset for six different values of Δt, comparing two key configurations: training and testing both with discretized spike times, versus training with continuous spike times and testing with discretized ones. Our analysis focuses on discrete testing, as this scenario reflects realistic deployment on hardware with fixed clock cycles. Although the network can be trained with high temporal precision in continuous time, inference is often constrained to run on discretized, clock-based hardware. By focusing on this combination, continuous-time training versus discrete-time deployment, we aim to assess whether aligning the discretized nature of spike representation across training and testing can improve both performance and sparsity.



(a) MNIST



(b) Extended MNIST



(c) Fashion MNIST

Fig. 6. Test accuracy (y-axis) versus average hidden-layer spike count (x-axis) for (a) MNIST, (b) Extended MNIST, and (c) Fashion MNIST. Each point represents the mean over 5 runs at a specific Δt (indicated by color). Circles correspond to networks trained in continuous time and tested with discretized spike times (C/D), while crosses denote networks trained and tested with discretized spike times (D/D). Across all datasets, consistent discretized training and inference (D/D) generally matches or slightly improves accuracy while substantially reducing spike counts, suggesting an efficiency benefit to aligning the discretization scheme.

Table 2. For a representative Δt, comparison of D/D (discretized training and testing) versus C/D (continuous training, discretized testing). D/D consistently achieves substantially lower spike counts while maintaining nearly identical accuracy, highlighting that matching the discretization scheme across training and inference can provide practical efficiency gains.

Dataset	Δt	Acc. (%)	Spikes	Setting
MNIST	0.002	97.35	∼255	D/D
	0.002	97.40	∼395	C/D
Ext. MNIST	0.002	78.0	∼550	D/D
	0.002	78.5	∼880	C/D
Fashion MNIST	0.00205	89.5	∼740	D/D
	0.00205	89.6	∼870	C/D

The results in Fig. 6 indicate that, for all datasets, networks trained and tested consistently with discretized spike times (crosses) generally achieve equal or higher test accuracy compared to networks trained in continuous time but evaluated with discretized spike times (circles). Minor exceptions occur at the largest Δt values, particularly for MNIST and Fashion MNIST. Importantly, Fig. 6 and Table 2 also show that models trained and tested under a consistent discretized regime exhibit substantially lower spike counts than their continuously trained counterparts. While the mechanisms underlying this reduction in spiking activity merit further exploration, the consistent pattern across datasets and time steps provides a practical guideline: aligning the discretization scheme between training and inference can yield meaningful efficiency gains with minimal impact on accuracy.

4 Conclusion and Discussion

In this work, we analyzed the effects of spike time discretization in spiking neural networks that operate with continuous-time dynamics during training but are deployed with discrete-time inference, as is often required by real-world neuromorphic hardware. We introduced a causal discretization scheme and characterized the induced quantization error, both analytically and empirically. Our experiments demonstrated that the degree of discretization has a non-trivial effect on both classification accuracy and network sparsity. We observed that aligning training and inference regimes, by training directly in discrete time, can yield similar or improved performance and reduced spiking activity, especially at coarser time resolutions. This consistency appears to help networks better adapt to the quantization introduced by discrete-time deployment. However, our results also indicate that these effects are not uniform across datasets: while improvements in accuracy and sparsity were generally observed, they were less pronounced or inconsistent at high Δt values for simpler datasets like MNIST.

It is important to note that our study focused on static image datasets converted into spike times through intensity-based encoding. In tasks where temporal information plays a central role, such as speech, audio, or event-based vision, the effects of discretization may be more significant, as spike timing becomes a critical component of information representation. Extending this analysis to such temporally structured datasets is an important direction for future work. Another consideration is the variability of discretization effects across datasets and architectures. Each dataset responded differently to discretization regimes, highlighting the need for broader evaluations and exploration of architectural or training objective variations to generalize our conclusions. Overall, our results emphasize that spike time discretization is not merely a technical necessity for hardware deployment but an important design parameter. Careful selection of discretization schemes can influence both the efficiency and performance of SNN models, providing a practical guideline for aligning training and inference in real-world applications.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Abbott, L.F.: Lapicque's introduction of the integrate-and-fire model neuron (1907) (1999)
2. Akopyan, F., et al.: TrueNorth: design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **34**(10), 1537–1557 (2015). https://doi.org/10.1109/TCAD.2015.2474396
3. Bacho, F., Chu, D.: Exploring tradeoffs in spiking neural networks. Neural Comput. **35**(10), 1627–1656 (2023). https://doi.org/10.1162/neco_a_01609
4. Bengio, Y., Léonard, N., Courville, A.C.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013)
5. Bohte, S.M., Kok, J.N., La Poutré, H.: Error-backpropagation in temporally encoded networks of spiking neurons. Neurocomputing **48**(1-4), 17–37 (2002). https://doi.org/10.1016/S0925-2312(01)00658-0, https://homepages.cwi.nl/~sbohte/publication/esann.pdf
6. Cohen, G., Afshar, S., Tapson, J., van Schaik, A.: EMNIST: an extension of MNIST to handwritten letters. arXiv preprint arXiv:1702.05373 (2017)
7. 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
8. Diehl, P.U., Cook, M.: Unsupervised learning of digit recognition using spike-timing-dependent plasticity. Front. Comput. Neurosci. **9**, 99 (2015). https://doi.org/10.3389/fncom.2015.00099
9. Gerstner, W., Kistler, W.M.: Spiking Neuron Models: Single Neurons, Populations. Cambridge University Press, Plasticity (2002)
10. Gish, H., Pierce, J.: Asymptotically efficient quantizing. IEEE Trans. Inf. Theory **14**(5), 676–683 (1968)

11. Goodman, D.M., Brette, R.: Brian: a simulator for spiking neural networks in python. Front. Neuroinform. **2**, 5 (2008). https://doi.org/10.3389/neuro.11.005.2008
12. Gütig, R., Sompolinsky, H.: The tempotron: a neuron that learns spike timing-based decisions. Nat. Neurosci. **9**(3), 420–428 (2006)
13. Göltz, J., Kriener, L., Baumbach, A., et al.: Fast and energy-efficient neuromorphic deep learning with first-spike times. Nat. Mach. Intell. **3**, 823–835 (2021). https://doi.org/10.1038/s42256-021-00388-x
14. Hunsberger, E., Eliasmith, C.: Spiking deep networks with LIF neurons (2015). http://arxiv.org/abs/1510.08829
15. Indiveri, G., et al.: Neuromorphic silicon neuron circuits (2011). https://doi.org/10.3389/fnins.2011.00073
16. Kempter, R., Gerstner, W., van Hemmen, J.L.: Hebbian learning and spiking neurons. Phys. Rev. E **59**(4), 4498 (1999)
17. Krizhevsky, A.: Learning multiple layers of features from tiny images (2009). https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf
18. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
19. Lichtsteiner, P., Posch, C., Delbruck, T.: A 128 × 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. IEEE J. Solid-State Circ. **43**(2), 566–576 (2008)
20. Maass, W.: Networks of spiking neurons: the third generation of neural network models (1997)
21. Maass, W., Markram, H.: On the computational power of circuits of spiking neurons. J. Comput. Syst. Sci. **69**(4), 593–616 (2004). https://doi.org/10.1016/j.jcss.2004.04.001
22. Neftci, E.O., et al.: Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks. IEEE Signal Process. Mag. **36**, 51–63 (2019)
23. Pehle, C., et al.: The BrainScales-2 accelerated neuromorphic system with hybrid plasticity. Front. Neurosci. **16**, 795876 (2022). https://doi.org/10.3389/fnins.2022.795876
24. Sengupta, A., Ye, Y., Wang, R., Liu, C., Roy, K.: Going deeper in spiking neural networks: VGG and residual architectures. Front. Neurosci. **13**, 95 (2019). https://doi.org/10.3389/fnins.2019.00095
25. Thorpe, S., Delorme, A., Van Rullen, R.: Spike-based strategies for rapid processing. Neural Netw. **14**(6–7), 715–725 (2001)
26. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
27. Yamazaki, K., Vo-Ho, V.K., Bulsara, D., Le, N.: Spiking neural networks and their applications: a review (2022). https://doi.org/10.3390/brainsci12070863
28. Zenke, F., Ganguli, S.: SuperSpike: supervised learning in multilayer spiking neural networks. Neural Comput. **30**(6), 1514–1541 (2018)
29. Zenke, F., Vogels, T.P.: The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. Neural Comput. **33**(4), 899–925 (2021). https://doi.org/10.1162/neco_a_01367