ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

# On the Importance of Time Constants in Spiking Neural Networks

Filippa Brandt[1,2], Saeed Bastani[1], Alexander Hunt[1], Amir Aminifar[2], Baktash Behmanesh[2] *

[1]Device Platform Research, Ericsson Research, Lund, Sweden

[2]Electrical and Information Technology, Lund University, Lund, Sweden

**Abstract**.

Time constants in spiking neural networks (SNNs) are crucial for determining performance. While prior work shows that learning time constants can improve accuracy, it typically assumes near-optimal initial values and rarely examines recovery from poor initializations. We systematically study how membrane and synaptic time constants affect SNN performance using multiple training strategies. Our results show that suboptimal values for time constants can reduce accuracy by nearly 10%, but networks can recover through optimization during the training process.

## 1 Introduction

Spiking neural networks (SNNs) process information over time by emitting discrete binary events, or spikes. How these spikes are integrated depends on the internal dynamics of neurons, in particular the membrane and synaptic time constants that govern how membrane potential and input current evolve. These time constants act as a form of short-term memory, determining the decay rate of past inputs, and typically operate on the 1-100 ms timescale [1]. As these parameters govern the temporal integration and neuronal responsiveness, optimizing them is essential for synchronizing the network's internal dynamics with the specific temporal granularities of a given task.

Time constants are predominantly treated as fixed hyperparameters, chosen based on biological plausibility. An alternative approach, however, is to learn them from data by training them jointly with synaptic weights using backpropagation through time (BPTT) and auto-differentiation. This idea has been explored in several works: for instance, [2] learn a layer-wise decay parameter for the membrane potential; [3] optimize neuron-wise membrane time constants; [4] train layer-wise membrane time constants reparameterized as their inverse and passed through a sigmoid; [5] learn heterogeneous neuron-wise decay factors; and [6] train both membrane and adaptation time constants in recurrent SNNs on sequential tasks.

These studies show that learning time constants generally improves performance over values. However, they typically initialize time constants within a narrow range that already yields reasonable accuracy, and rarely ask whether

*This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation and the Swedish Foundation for Strategic Research.

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

the learned values are truly optimal or whether models can recover from poor initializations. Moreover, synaptic time constants are often omitted [3] or restricted to a single global or layer-wise value rather than neuron-specific values [4]. This leaves an open question: is learning time constants beneficial when a reasonable range cannot be pre-determined, and how robust training is to suboptimal initial choices?

To address these gaps, we perform a grid search over membrane ($\tau_m$) and synaptic ($\tau_s$) time constants to map accuracy across the parameter space and evaluate how these parameters affect performance, which in turn provides us with an intuition of optimal ranges of $\tau$ for a given dataset. We compare two discretization schemes for the neuron dynamics, evaluate three reparameterizations of the time constants, and test whether SNNs training strategies for time constants can recover when starting from suboptimal values. Our results show that optimal time constant values exist, can be learned, and that their choice, even within commonly used ranges, has a substantial impact on model accuracy.

## 2 Method

To investigate the role of time constants on SNN performance, we perform a systematic study based on the Leaky Integrate-and-Fire (LIF) neuron model. First, we discuss two state-of-the-art discretization schemes: Euler integration and exponential decay, each resulting in different reparameterizations for learning time constants. Next, we analyze the performance landscape of SNNs across time constant values using a grid search and investigate whether networks can recover from suboptimal time constant initializations.

A LIF neuron $i$ is governed by its membrane potential [1]:

$$\dot{v}_i(t) = \frac{1}{\tau_m}\big(v_{\text{leak}} - v_i(t) + i_i(t)R\big), \tag{1}$$

where $\tau_m$ is the membrane time constant, $v_{\text{leak}}$ is the leak potential, $R$ is the membrane resistance, and $i(t)$ is the synaptic input current. For simplicity, we set $R = 1$ and $v_{\text{leak}} = 0$. In both biological and computational models, synaptic input does not influence the membrane potential instantaneously; instead, it evolves gradually over time. The synaptic input current is therefore described by the following temporal dynamics:

$$\dot{i}_i(t) = -\frac{1}{\tau_s} i_i(t) + \sum_j W_{ij} z_j(t), \tag{2}$$

where $\tau_s$ is the synaptic time constant, $W_{ij}$ are the synaptic weights between neuron $i$ and $j$, and $z_j(t)$ are the spike trains from pre-synaptic neurons. When the membrane potential $v(t)$ reaches a threshold $v_{th}$, the neuron emits a spike and the membrane voltage is reset to $v_0$. Consistent with prior work [5], [7], [3], we assume $v_{th} = 1$ and $v_0 = 0$.

These continuous-time equations must be converted into discrete form, and there are several ways to do so. In the following, we examine two discretization

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

methods, each leading to distinct reparameterizations for learning time constants.

**Euler Integration**. Using Euler discretization, following [7], the incoming current is added first to form $i_{new}$, and the decay is applied later:

$$i_{\text{new}} = i_i[n] + \sum_j W_{ij} z_j[n]$$

$$v_i[n+1] = v_i[n] + \frac{\Delta t}{\tau_m}\Big( - v_i[n] + i_{\text{new}}\Big) \tag{3}$$

$$i_i[n+1] = i_{\text{new}} \left(1 - \frac{\Delta t}{\tau_s}\right), \tag{4}$$

where $n$ denotes the discrete time step and $\Delta t$ is the time step size. To train the time constants, we consider two parametrization strategies. One approach is to set $\alpha = 1/\tau_m$ and $\beta = 1/\tau_s$, which is the main strategy in [7]. The other is our adaptation defining $\alpha = \Delta t/\tau_m$ and $\beta = \Delta t/\tau_s$ to avoid gradient scaling issues caused by small time steps.

**Exponential Decay Integration**. An alternative approach is to solve the linear ODEs exactly. Following [5], this yields:

$$i_i[n+1] = \beta \cdot i_i[n] + \sum_j W_{ji} z_i[n] \tag{5}$$

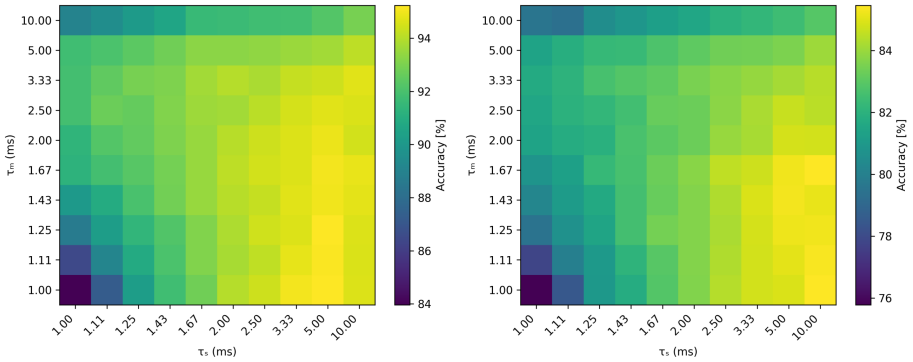
$$v_i[n+1] = \alpha \cdot v_i[n] + (1-\alpha) \cdot i_i[n], \tag{6}$$

where $\alpha = e^{-\Delta t/\tau_m}$ and $\beta = e^{-\Delta t/\tau_s}$.

We begin our investigation by examining the effect of network-wide shared time constants, where every neuron shares the same $\tau_m$ and $\tau_s$. For the Euler discretization, we sample ten values with $\tau_m, \tau_s \in [\Delta t, 10]$ ms. For the exponential-decay method, we sample ten values logarithmically with $\tau_m, \tau_s \in [\Delta t, 100]$ ms to cover several orders of magnitude more effectively. This ensures that $\alpha$ and $\beta$ remain in the interval [0,1], which helps maintain numerical stability. We then evaluate all combinations of $\tau_m$, $\tau_s$ for both approaches. This grid search reveals which regions of the parameter space yield the best performance for fixed time constant values. Next, we examine whether the network can recover from a sub-optimal initialization of the time constants by learning their parameterization during the training process and converging to a point in the optimal region in the grid.

Using the results of the grid search, we initialize with time constants that corresponds to the lowest inference accuracy, and explore two recovery settings: In the first, time constants remain shared within each layer throughout training. In the second, all neurons start with the same sub-optimal initialization but learn their time constants independently. $\alpha$ and $\beta$ are initialized according to each reparameterization using these sub-optimal values. Both experiments are repeated for each discretization method. For stability, $\alpha$ and $\beta$ are constrained to the interval [0, 0.995] after every update. During training, $\alpha$ and $\beta$ function as ordinary learnable parameters and are optimized together with the synaptic weights.

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.




(a) MNIST (b) F-MNIST

Fig. 1: Performance across different combinations of fixed $\tau_m$ and $\tau_s$ with our adaptation of Norse [7], for MNIST [8] and Fashion-MNIST (F-MNIST) [9].

## 3 Experimental Setup and Results

We evaluate our approach on MNIST [8] and Fashion-MNIST (F-MNIST) [9] with Poisson-encoded input images. Experiments are implemented in PyTorch, using components from Norse [7], including the Poisson encoder, SuperSpike [10] surrogate gradient, and LIF neuron implementation. The network is a simple feedforward SNN with a single hidden layer of 20 neurons. Both input and hidden layers use LIF neurons. The output layer uses non-spiking leaky-integrator (LI) neurons that accumulate inputs without resetting. The network output is decoded via peak voltage: for each output neuron, we take its maximum membrane potential across time and then apply a softmax across these peak values to obtain class probabilities, i.e., $\hat{y} = \text{softmax}\Big(\max_t(V[t])\Big)$. We train using Adam with learning rate 0.001 for both weights and time constants. When training time constants, $\alpha$ and $\beta$ are defined as trainable parameters in PyTorch, with gradients computed via automatic differentiation. The simulation length is set to $5\tau_{max}$ to ensure network dynamics fully express the effect of the largest time constants, resulting in $10\times$ shorter simulation time and reduced training complexity for Euler integration approaches.

Figure 1 shows the performance across different fixed $\tau_m$ and $\tau_s$ combinations using our Norse adaptation (NOx). Each square shows the mean accuracy over three runs. The results indicate that the choice of time constant values can substantially impact performance, with differences of nearly 10%.

Table 1 compares three SNN learning schemes to train time constants, i.e., Neural Heterogeneity (NH) [5], the original Norse training (NO) [7], and our adaptation of Norse training (NOx), on MNIST [8] and Fashion-MNIST [9]. The results clearly indicate that time constants are crucial parameters that must be explicitly considered and learned during training. For both datasets, the NH and NOx methods demonstrate robustness to poor initialization with performance

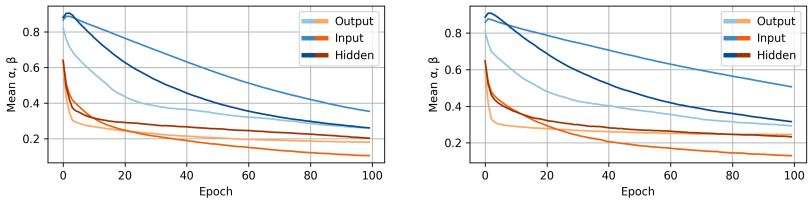
ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

Table 1: Comparison of three SNN training schemes: "Fixed" denotes untrained time constants, "Layer-wise" denotes layer-wise shared learned time constants, and "Neuron-wise" denotes neuron-wise learned time constants.

| **Training Strategy** | **Suboptimal** $(\tau_m, \tau_s)$ | **Accuracy for MNIST (%)** | | | **Accuracy for F-MNIST (%)** | | |
|---|---|---|---|---|---|---|---|
| | | Fixed | Layer-wise | Neuron-wise | Fixed | Layer-wise | Neuron-wise |
| NH [5] | (100 ms, 100 ms) | $76.75 \pm 7.27$ | $95.81 \pm 0.06$ | $95.77 \pm 0.18$ | $79.72 \pm 0.39$ | $86.54 \pm 0.45$ | $86.20 \pm 0.55$ |
| NO [7] | (10 ms, 1 ms) | $83.42 \pm 0.76$ | $84.66 \pm 0.63$ | $85.07 \pm 0.71$ | $52.19 \pm 2.10$ | $52.81 \pm 1.19$ | $57.05 \pm 4.00$ |
| NOx | (1 ms, 1 ms) | $83.97 \pm 0.21$ | $94.62 \pm 0.24$ | $94.70 \pm 0.24$ | $75.79 \pm 1.44$ | $84.30 \pm 0.09$ | $84.68 \pm 0.12$ |

improvement when starting from suboptimal time constant values. For instance, considering the MNIST dataset and NOx, the performance can be improved from $83.97 \pm 0.21$ for fixed mean values for the time constants, to $94.62 \pm 0.24$ and $94.70 \pm 0.24$ for optimized time constants, which represents an improvement of over 10%. For F-MNIST, NH achieves higher absolute accuracy (86% vs 84%) while NOx achieves a larger relative improvement (∼9% vs ∼7%). Furthermore, the negligible difference between layer-wise and neuron-wise training suggests that enabling time constant adaptation is more important than granularity for these small-scale tasks. The NO approach shows minimal improvement, likely due to the gradient scaling issue inherent in its reparameterization since $\alpha = 1/\tau_m$ and $\alpha = 1/\tau_s$ are multiplied by the small time step $\Delta t = 0.001$ (Eqs. 3 and 4), resulting gradients become small, slowing down learning.

Figure 2 shows how the mean values of $\alpha$ and $\beta$ across all layers evolve over three training runs when time constants are learned individually. Starting from short time constants (1 ms), corresponding to large $\alpha$ and $\beta$ values, both parameters generally decrease during training, indicating a shift toward longer time constants. This trend is consistent with the heatmap results, where the highest accuracy occurs for longer time constants. We also observe that $\alpha$ converges to slightly larger values than $\beta$, so $\tau_m$ remains shorter than $\tau_s$, again matching the optimal region in the heatmap.




(a) MNIST (b) F-MNIST

Fig. 2: Evolution of $\alpha$ and $\beta$ in input, hidden, and output layers during training for our method with individually trained time constants (i.e. Neuron-wise scenario). Blue and orange curves correspond to $\alpha$ and $\beta$, respectively.

ESANN 2026 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 22-24 April 2026, i6doc.com publ., ISBN 9782875870964. Available from http://www.i6doc.com/en/.

## 4 Conclusions

In this paper, we investigated how membrane and synaptic time constants influence network behavior by examining different reparameterizations and training granularities. Using state-of-the-art training schemes and standard benchmarks, we demonstrate that both the choice of reparameterization and the handling of time constants significantly affect model performance and highlight that reparameterization impacts computational efficiency, as different formulations can lead to substantially different simulation times. Suboptimal initialization can reduce accuracy by nearly 10%, although this loss can be recovered via time constant optimization during training. These findings highlight the importance of carefully handling time constants when designing SNNs. Although our experiments focused on relatively small models, the sizable performance variations observed indicate that time constant optimization is likely to be even more critical in larger architectures and real-world applications, e.g., on low-power wearables in epilepsy [11], making this an important direction for future work.

## References

[1] Jason K. Eshraghian, Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training Spiking Neural Networks Using Lessons From Deep Learning. *Proceedings of the IEEE*, 111(9):1016–1054, September 2023.

[2] Romain Zimmer, Thomas Pellegrini, Srisht Fateh Singh, and Timothée Masquelier. Technical report: supervised training of convolutional spiking neural networks with PyTorch.

[3] Filippo Moro, Pau Vilimelis Aceituno, Laura Kriener, and Melika Payvand. The role of temporal hierarchy in spiking neural networks.

[4] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating Learnable Membrane Time Constant to Enhance Learning of Spiking Neural Networks. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2641–2651, October 2021. ISSN: 2380-7504.

[5] Nicolas Perez-Nieves, Vincent C. H. Leung, Pier Luigi Dragotti, and Dan F. M. Goodman. Neural heterogeneity promotes robust learning. *Nature Communications*, 12(1):5791, October 2021.

[6] Bojian Yin, Federico Corradi, and Sander M. Bohté. Effective and Efficient Computation with Multiple-timescale Spiking Recurrent Neural Networks, June 2020. arXiv:2005.11633 [cs].

[7] Christian Pehle and Jens Egholm Pedersen. Norse - A deep learning library for spiking neural networks, January 2021. Documentation: https://norse.ai/docs/.

[8] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[9] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *ArXiv*, abs/1708.07747, 2017.

[10] Friedemann Zenke and Surya Ganguli. SuperSpike: Supervised learning in multilayer spiking neural networks. *Neural Computation*, 30(6):1514–1541, June 2018. arXiv:1705.11146 [cs, q-bio, stat].

[11] Xavante Erickson, Saeed Bastani, and Amir Aminifar. Personalized seizure detection using spiking neural networks. In *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, pages 1–6. IEEE, 2023.