Temporal hierarchy in spiking neural networks

NEUROMORPHIC
Computing and Engineering

PAPER

Temporal hierarchy in spiking neural networks

Filippo Moro*, Pau Vilimelis Aceituno, Laura Kriener and Melika Payvand*
UZH and ETH Zurich, Institute of Neuroinformatics, Zürich, Switzerland
* Authors to whom any correspondence should be addressed.
E-mail: filippo.moro@uzh.ch and melika.payvand@uzh.ch

OPEN ACCESS

RECEIVED
22 March 2026

REVISED
22 May 2026

ACCEPTED FOR PUBLICATION
9 June 2026

PUBLISHED
23 June 2026

Keywords: spiking neural networks, neuromorphic, time scale hierarchy, NeuroAI

Supplementary material for this article is available online

Abstract

Taking inspiration from the brain to design efficient computational systems remains challenging due to its complexity. This work investigates a key biological feature observed across mammals: a hierarchy of time scales in cortical areas. Experimental evidence shows that intrinsic neural dynamics slow down across the cortical hierarchy, forming a temporal hierarchy. We examine whether this property benefits artificial systems by introducing temporal hierarchy into spiking neural networks (SNNs), which inherently process information over time. We implement hierarchical time scales across neuronal, synaptic, and recurrent dynamics, and evaluate their effect under two settings: (1) as an inductive bias, and (2) as an emergent property through optimization. On temporal benchmarks such as multi-timescale-XOR and keyword spotting, hierarchical SNNs consistently outperform non-hierarchical ones, achieving 2%–6% higher accuracy and up to 5× parameter reduction under iso-accuracy conditions. Moreover, when trained freely, temporal hierarchy emerges spontaneously through gradient descent. Finally, our theoretical analysis shows that hierarchical time constants enable processing of multi-frequency temporal signals with only log N layers—compared to N layers for non-hierarchical systems—highlighting hierarchy as a key organizational principle for efficient temporal computation.

1. Introduction

Extrapolating computational primitives from the brain to the benefit of artificial systems is as fascinating as it is challenging. The brain is by far the most efficient known computing system, but some of the mechanisms underlying its efficiency remain mostly unknown. Moreover, identifying which specific features of the brain are most relevant and beneficial for artificial computational purposes is an open question. In this work, we propose to investigate a well-established feature of the mammalian cortex, the hierarchy of time scales in cortical areas, and demonstrate its utility for computational performance. Research has shown that the timescale of spiking activity increases along cortical hierarchies in primates [1–3]. Recent work [4] has studied the mechanisms driving these timescale gradients and their relevance to cognitive functions, while also investigating data analysis techniques to examine neural recordings and relate them to biological temporal computational primitives. Previous works have introduced *heterogeneity* of neuronal time constants in spiking neural networks (SNNs) [5, 6], emphasizing neuron and dendritic-level diversity as a means to enrich temporal feature representations. In contrast, our work introduces and formalizes temporal hierarchy as a structured progression of temporal processing speeds across layers. While within-layer heterogeneity and across-layer hierarchy are complementary, a key distinction is that hierarchy imposes a directional structure (fast-to-slow) whose computational benefits we formalize in this work. To the best of our knowledge, the reason why such a hierarchy of time scales is useful at a computational level has not yet been addressed. Thus, we examine whether incorporating this cortical feature into artificial systems provides computational benefits and explore how a hierarchy of timescales can be implemented through computational primitives.

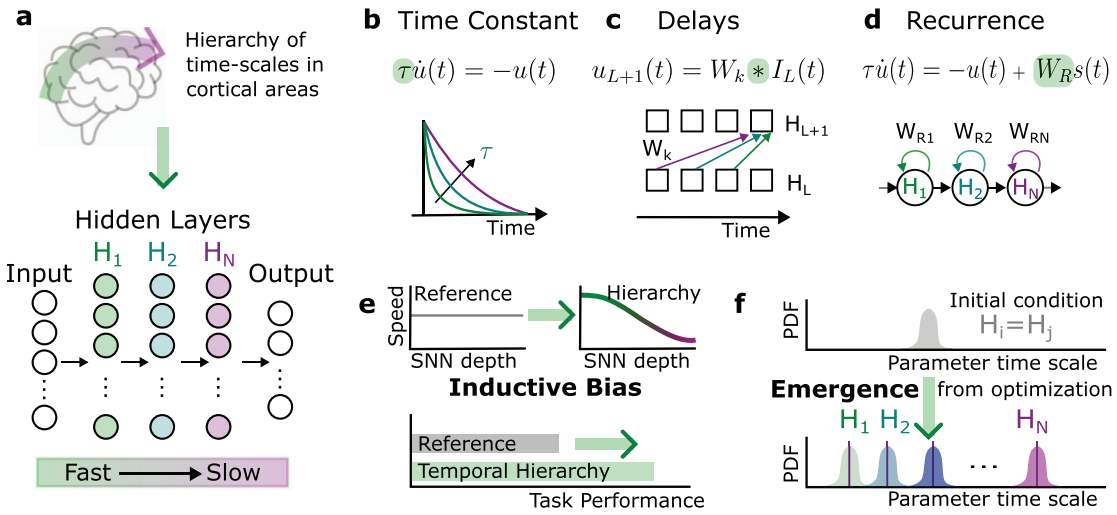

Figure 1. Hierarchy of time scales from the brain to artificial computing systems. (a) Temporal hierarchy is extrapolated from neuroscience findings and transferred to multi-layer SNNs featuring $\mathcal{N}$ hidden layers colored differently, highlighting the difference in temporal processing speed, from fast to slow through the layers. (b) Time constant $\boldsymbol{\tau}$ determining the speed of neuronal dynamics in leaky integrate and fire (LIF) spiking neurons. The differential equation describes the dynamics for the membrane voltage $u(t)$, which evolves with a time constant $\boldsymbol{\tau}$. (c) Delay expressed as temporal convolutions featuring a kernel W_k, affecting the state of neurons $u_{L+1}(t)$ in time. (d) Temporal processing in multi-layer recurrent spiking neural networks (SNNs), the speed of each layer is determined by the recurrent weights W_{Rl}. (e) Temporal hierarchy in the inductive bias setting: hierarchy is introduced in the temporal parameters of the multi-layer SNN at initialization and the performance is compared with a reference case where all layers possess the same processing speed. (f) Temporal hierarchy verified via emergence from optimization: multi-layer SNNs are initialized without any structure in the temporal parameters, and those are later optimized via gradient descent while solving a task.

To conduct this investigation, we utilize SNNs as our computational framework due to their biological inspiration and inherent temporal computation capabilities. Furthermore, SNNs have shown promise for real-time sensory processing applications at the edge [7], making our investigation transferable to practical applications. SNNs can be formulated and optimized as recurrent artificial neural networks (ANNs) [8], with the key constraint that neuronal communication occurs solely through binary signals (spikes). We specifically analyze the implications of imposing a hierarchy in the speed of temporal processing in SNNs. The temporal hierarchy is implemented as a configuration where the speed of temporal processing in a multi-layer SNN progressively slows down across hidden layers, transitioning from faster to slower dynamics (figure 1(a)). To evaluate the impact of this hierarchy, we explore SNNs with various temporal parameters, including the 'speed' of neuronal dynamics (i.e. neuronal time constant, figure 1(b)) [9], delay parameters in the spike transmission [10, 11] or temporal causal convolutions [12] (figure 1(c)), and recurrent connections resulting in short-term memory [13, 14] (figure 1(d)).

Our experiments are categorized into two settings. (1) In the 'inductive bias' setting (figure 1(e)), SNNs are initialized with a predefined aforementioned temporal hierarchy, while all other parameters are optimized via gradient descent. (2) In the 'Emergence from optimization' setting (figure 1(f)), the temporal parameters of SNNs are independently initialized from the same distribution across all hidden layers and subsequently optimized through gradient descent.

We test the networks in these settings, under three benchmark classification tasks, multi-time-scale temporal XOR (MTS-XOR), spoken digits classification (spiking Heidelberg digit (SHD)), and spoken commands recognition (spiking speech command (SSC)). In the inductive bias setting, we demonstrate an improvement in classification accuracy when we introduce a temporal hierarchy in SNNs through neuronal time constants (figure 1(b)) and synaptic delays (figure 1(c)). Specifically, there is a gain of 6%, 3.8%, and 2.1% when testing on the MTS-XOR, SHD, and SSC tasks, respectively. Notably, such accuracy gain is more prominent under settings with a lower number of parameters. Moreover, we demonstrate that when targeting a certain level of performance, introducing hierarchy reduces the required parameter count to reach such target by 2× and 5× in the MTS-XOR and SHD tasks respectively. In the emergence from optimization setting, we observe that gradient descent naturally discovers temporal hierarchies, both in neuronal time constants and recurrent dynamics (figure 1(d)).

In addition to our simulation results, we develop a mathematical framework to analyze the computational capacity of SNNs using principles from signal processing. By modeling the computation of spiking

neurons as information filters, we show that the range of temporal patterns a single neuron can recognize is inherently limited. This limitation highlights the necessity of diverse neural time scales for processing complex sequences and demonstrates that organizing these timescales in a hierarchical structure can overcome such limitations, while substantially reducing the number of required layers.

Overall, our findings highlight the value of temporal hierarchy as a beneficial architectural feature for SNNs and offer a straightforward heuristic for enhancing their performance in temporal tasks. This work has important implications for resource-constrained computation and edge artificial intelligence, where efficiency in parameter use is critical.

2. Results

Here, we demonstrate the effect of temporal hierarchy under the two settings of inductive bias and emergence from optimization. In both cases, the neurons in the SNNs are modeled using the widely-adopted leaky integrate and fire (LIF) neuron model [15]: a first-order differential equation (figure 1(b)) governs its state variable $u(t)$, often called membrane voltage, which integrates the incoming spikes, and once it hits a threshold, the neuron elicits a spike.

In the first series of experiments, we investigate the role of the LIF neuronal dynamics, thus considering feed-forward SNNs where the sole temporal computational primitive is the membrane voltage dynamics, i.e. an exponential decay with time constant τ. Such dynamics determine the 'speed' of LIF neurons and act as a temporal feature detector [16, 17], and filter [18]. To generalize the effects of the temporal hierarchy beyond membrane time constants, to different temporal computational primitives, we later add synaptic delay parameters (temporal convolution) and recurrent connections to SNNs. For more details on the SNNs implementation, refer to the Methods section 4.1.

2.1. Time constant hierarchy as inductive bias in feed-forward SNN

In this section, we evaluate the effect of temporal hierarchy as an inductive bias, using feed-forward SNNs, where the key temporal parameter is the neuron time constant τ. We define the temporal hierarchy of time constants as the initialization of such parameters from fast in the first layer and progressively slower in the deeper layers, such that with H_l hidden layers, the time constants at layer l follows:

$$\tau(l) = \tau_\mu + (l - H_l/2)\,\Delta\tau/2.$$

To isolate the effect of the temporal hierarchy, we first consider SNNs with a homogeneous time constant τ_μ through the layers, optimizing such value as a hyperparameter. After finding the optimal homogeneous τ_μ, we alter the time constant across layers by an amplitude $\Delta\tau$ and compare the performances for different amplitude changes. In terms of architecture, we utilize either 2 or 3 hidden layers with a size of 10 neurons for the MTS-XOR tasks. In keyword spotting tasks, the size varies from 2 to 5 hidden layers, with a hidden layer size of 32 or 128 neurons.

In our theoretical analysis (see section SI.F.2) we demonstrate that a long enough time constant can optimize performance in tasks such as MTS-XOR. This hypothesis is verified in figure SI.2(a), where it is clear that classification accuracy on this task saturates when the time constant is increased past a certain threshold ($\tau > 300$ ms). We thus demonstrate that a further performance improvement on this task cannot be obtained by simply slowing down neuronal dynamics, but a more sophisticated temporal processing is necessary. To further investigate the role of time constants, we evaluate performance on the more complex keyword-spotting task using the SHD dataset [19]. We employ SNNs with three hidden layers, each containing 128 neurons, and a uniform time constant across all layers. We find out that the SHD task demands different network characteristics than the MTS-XOR task: as shown in figure SI.2(b), performance in homogeneous SNNs is maximized at $\tau_\mu = 100$ ms beyond which accuracy drops. This preliminary analysis sets the *reference* SNNs with an optimal homogeneous time constant of $\tau_\mu = 300$ ms for MTS-XOR and $\tau_\mu = 100$ ms for keyword spotting (SHD and SSC) tasks. In the following, we endow the reference SNNs with temporal hierarchy, modifying the time constant at each layer $\tau(l)$ to create a hierarchy of time constants proportional to $\Delta\tau$, i.e. the time constant difference across the hidden layers.

To analyze the effects of temporal hierarchy, we construct three experiments that are performed on both the MTS-XOR and the SHD tasks. To study the effect of temporal hierarchy on the accuracy, (i)

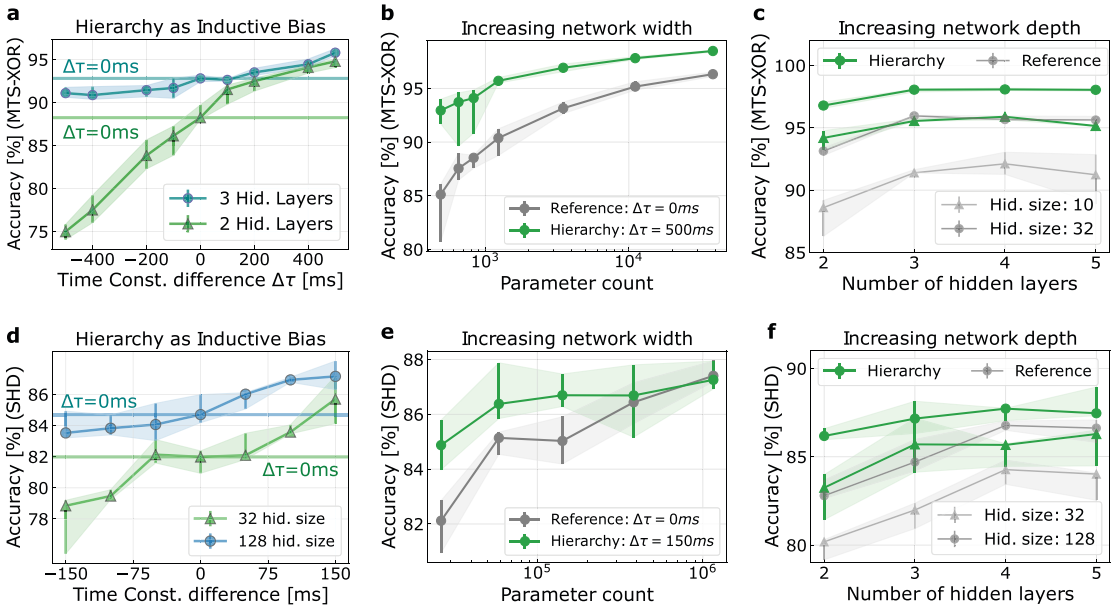

Figure 2. Temporal hierarchy in the time constants of SNNs. (a) Classification accuracy when temporal hierarchy is introduced in multi-layer SNNs with varying amplitude ($\Delta\tau$: time constant difference across hidden layers). Positive $\Delta\tau$ improves the classification accuracy. (b) Effect of temporal hierarchy when increasing network width on the MTS-XOR task. A 2-hidden-layers SNN is used with a number of hidden neurons in each layer ranging from 8 to 256 neurons (network width). (c) Effect of temporal hierarchy with increasing network depth on the MTS-XOR task. SNNs with 10 or 32 hidden neurons per layer, with and without temporal hierarchy, are progressively made deeper (from 2 to 5 hidden layers). (d) Temporal hierarchy benchmarked on the SHD task. Two network sizes are considered, featuring 3 hidden layers, with a hidden layer size of 32 or 128. The smaller network is more sensitive to the hierarchy of time constants. Horizontal lines in (d) refer to the performance in the no hierarchy case (reference), i.e. $\Delta\tau = 0$ ms. (e) Effect of temporal hierarchy when increasing network width on the SHD task, where 3-hidden layer SNNs with width from 32 to 512 neurons are tested. (f) Effect of temporal hierarchy with increasing network depth on the SHD task. SNNs with 32 or 128 hidden neurons per layer, with and without temporal hierarchy, are progressively made deeper (from 2 to 5 hidden layers).

the amplitude of temporal hierarchy $\Delta\tau$ is varied ranging from negative (the opposite of temporal hierarchy) to positive values; To study the effect of temporal hierarchy on the number of parameters, temporal hierarchy is compared against the *reference* case when sweeping (ii) the width of the SNNs layers; and (iii) the depth of the SNNs.

MTS-XOR task Experiment (i) results in figure 2(a) showing that the classification accuracy positively correlates with the hierarchy amplitude $\Delta\tau$. We hypothesize that this is due to the functional role of different layers in a multi-layer SNN, with the first hidden layer acting as a feature extractor, while the deeper hidden layers integrate information over longer time spans [20]. Moreover, we observe that the temporal hierarchy with positive $\Delta\tau$ results in a sharper gain in classification accuracy in the shallower SNNs (2 hidden layers (green) vs 3 hidden layers (blue)). This indicates that the inductive bias is more relevant under resource constraints, i.e. available parameter count.

To study this observation further, we perform experiments (ii) and (iii). Sweeping the network width, we train 2-hidden-layer SNNs with hidden neuron sizes varying from 8 to 256 neurons. Figure 2(b) shows the network accuracy as a function of its width with (green) and without (gray) the imposed temporal hierarchy. Notably, SNNs trained with a temporal hierarchy as inductive bias always perform better than the ones without the hierarchy (reference case), but the performance gap shrinks when the network width is increased. Sweeping the network depth from 2 to 5 layers, we test the accuracy of the SNNs with 10 neurons and 32 neurons per layer. The accuracy results in figure 2(c) indicates that the temporal hierarchy provides a performance benefit across all network depths, which persists even when SNNs feature enough hidden layers to saturate the task performance.

SHD task Experiment (i) is repeated on the SHD task, as shown in figure 2(d), confirming the correlation between classification accuracy and temporal hierarchy amplitude. Interestingly, the SNNs with 32 neurons per hidden layer with temporal hierarchy $\Delta\tau = 150$ ms performs better than the larger SNNs with 128 neurons per hidden layer and homogeneous time constants ($\Delta\tau = 0$ ms, blue horizontal line). In other words, the network endowed with hierarchy performs better than the network without hierarchy, while using 5 times fewer number of parameters.

In experiment (ii), we train 3-hidden-layer SNNs with hidden layer size changing from 32 to 512 neurons. Figure 2(e) shows that the performance advantage induced by temporal hierarchy shrinks when the network width increases. Sweeping the network depth from 2 to 5 layers (experiment (iii)), we train SNNs in the two cases of a network width of 32 and 128 neurons. Figure 2(f) shows that the temporal hierarchy improves the performance for both shallow and deep SNNs, although the classification accuracy advantage decreases with network depth. This analysis of the network size across its width and depth strengthens the conclusion on the higher impact of temporal hierarchy in settings where a lower number of memory elements are available, e.g., in edge AI applications.

Furthermore, the hierarchy of time constants is tested on the SSC dataset [21], showing a performance boost compared with a network with homogeneous time constants. Table 1 reports the summary of the results of our temporal hierarchy-endowed SNNs on the SHD and SSC datasets.

Disentangling temporal hierarchy and heterogeneity A natural question is whether the performance gains stem from the structured progression of time constants across layers (hierarchy) or simply from having diverse time constants (heterogeneity), as in [5]. To disentangle these effects, we jointly vary the width of the intra-layer time constant distribution and the hierarchy amplitude $\Delta\tau$ (figure SI.3). The results show that hierarchy is the dominant factor: the accuracy difference between hierarchical and non-hierarchical conditions consistently exceeds the effect of varying heterogeneity alone. Increasing heterogeneity provides a modest benefit on SHD but is negligible on MTS-XOR. Importantly, hierarchy and heterogeneity are complementary. The best performance is achieved when both are present at the same time.

The optimal shape of temporal hierarchy So far, temporal hierarchy is imposed as the difference of time constants through the hidden layers of the SNNs, where the values of time constants are uniformly spaced in the network depth. We investigate the optimal progression of time constants—which we refer to as the 'shape' of hierarchy—through the hidden layers of the SNN in appendix SI.B. We describe the time constant at a particular hidden layer by a hyperbolic-tangent function taking a steepness (**s**) and a centering (**c**) parameter, allowing for different profiles of temporal hierarchy, looking for the combination of the parameters that yield higher performance in the keyword-spotting task. The results in figure SI.4 show that any combinations of centering and steepness provide an accuracy increment compared to the reference (without temporal hierarchy), thus temporal hierarchy is a positive asset independently of the "shape" of time constants, as long as these are getting slower through the hidden layers. Certain shapes of the temporal hierarchy are more beneficial and those are reported in figure SI.4(c): despite not testing the hypothesis further, we propose that the optimal shape of temporal hierarchy is task-dependent.

Temporal hierarchy with non-temporal tasks To ensure that the advantages of the temporal hierarchy emerge from the requirements of temporal tasks, we also tried tackling a non-temporal task (static MNIST digits), finding that the temporal hierarchy has no significant effect in this setting (see figure SI.5 in appendix SI.C). As the pixel brightness of static images is converted to spike latency in this setting, the task performance can theoretically be optimized by integrating information over time. It is not surprising that infinite time constants are desirable in this context and that arranging them in a temporal hierarchy results in no practical benefit. However, turning this static task into a temporal task by encoding the input in a sequential manner (sequential MNIST [22]), the temporal hierarchy brought a significant improvement in classification accuracy (see figure SI.6). In this setting, information is accessed one pixel at a time, thus forming a memory of the past seen pixel is crucial for classification performance. However, setting the time constant too high leads to 'overwriting' the neurons' formed memory, losing the sense of the sequence. Here, temporal hierarchy helps balance memory formation and maintain selectivity to sequences, improving classification performance.

2.2. Time constant hierarchy emerges from optimization

The experiments in the previous section demonstrate the effectiveness of using temporal hierarchy in the initialization of LIF neuron time constants as an inductive bias for SNNs. However, in those experiments, the time constants were not optimized. In this section, we explore the outcomes of optimizing the time constants of LIF neurons in SNNs, to identify emergent patterns in the trained values. We initialize SNNs with a neuron time constant drawn from a Gaussian distribution with a given mean (τ_μ) and standard deviation of 20% of the mean. Such initialization is consistent across all the hidden layers in the SNNs. Using gradient descent, we update the time constant of hidden neurons while training

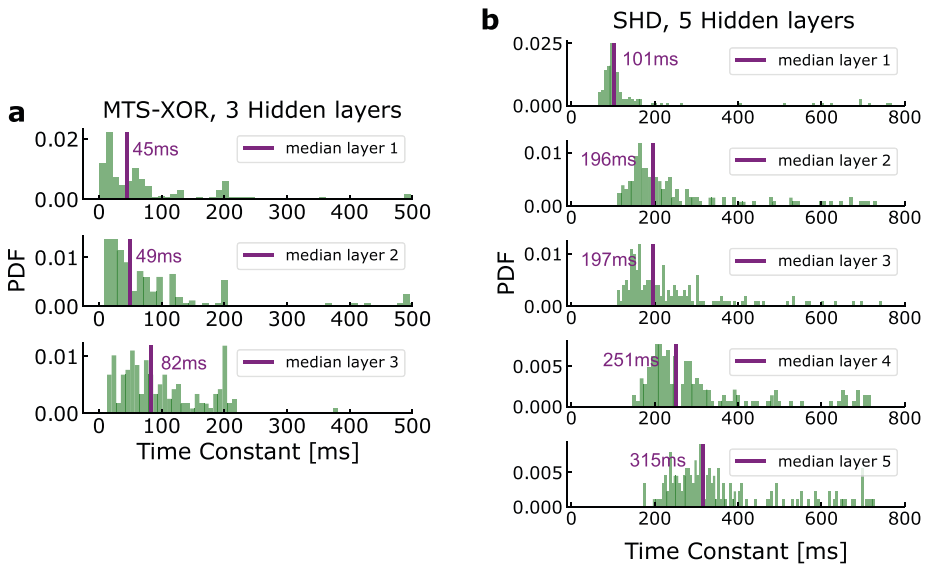

Figure 3. Temporal hierarchy emerging from optimization of the SNNs time constants. Multi-layer SNNs are initialized with the same distribution of time constants through the hidden layers. Each time constant is optimized individually. (a) Probability density function for the optimized time constants in a 3-hidden-layer SNN solving the MTS-XOR task. The median time constant (purple lines) grows in each layer, indicating the formation of a hierarchy. (b) Median time constant (purple) from 5-hidden-layer SNNs solving the SHD task. The median grows through the hidden layers indicating the hierarchy emerged in the network. Results are averaged from 5 trials in both (a) and (b).

the SNNs to solve the MTS-XOR and SHD tasks (figure 1(e)). For more details on the optimization of time constants, refer to the Methods (section 4.1). The probability density function of the time constant in the hidden layer of 3 SNNs trained on the MTS-XOR is shown in figure 3(a). The medians of the time-constant distributions grow progressively throughout the SNN layers. This confirms that the optimization algorithm also discovers the hierarchy of time constants.

We repeated the experiment with 5-hidden layer SNNs trained on the SHD task (figure 3(b)). We show that the resulting median values of the time constant distributions grow through the hidden layer, further demonstrating that optimization leads to the emergence of a temporal hierarchy.

2.3. Extending temporal hierarchy in SNNs to temporal causal convolution and recurrent connectivity

We have thus far applied the concept of temporal hierarchy to neuronal time constants. Temporal hierarchy in SNNs can also be expressed through other temporal parameters. In this section, we examine the impact of applying temporal hierarchy to temporal causal convolutions and recurrent connectivity. For temporal convolutions, we introduce temporal hierarchy as an inductive bias and analyze its effects. For recurrent connectivity, we optimize the network and assess whether a hierarchy of temporal processing naturally emerges.

Temporal hierarchy as an inductive bias in temporal causal convolutions Temporal causal convolutions or delays have been demonstrated to be effective in improving the expressiveness of SNNs [10–12, 23–25].

The dilation in temporal convolution determines the receptive field size of post-synaptic neurons, as shown in figure 4(a). As temporal convolutions effectively present inputs to a post-synaptic neuron with temporal shift—modulated by the dilation or kernel size—they are effectively equivalent to synaptic delays. With a fixed kernel size, larger dilation allows post-synaptic neurons to access inputs from further back in time, while smaller dilation limits them to immediate past inputs. Importantly, changing dilation does not affect the number of trainable parameters, but only the size of a post-synaptic neuron's receptive field.

Here, we construct temporal hierarchy by progressively increasing dilation through the layers of the SNN (figure 4(a)), denoting the dilation difference across the network as Δ_{dil}. To isolate the effect of dilations, we set the neurons' time constants from 100 ms to lower values (20 ms). We then perform experiments with 2-hidden-layer SNNs with 128 neurons per hidden layer, optimized to solve the SHD task.

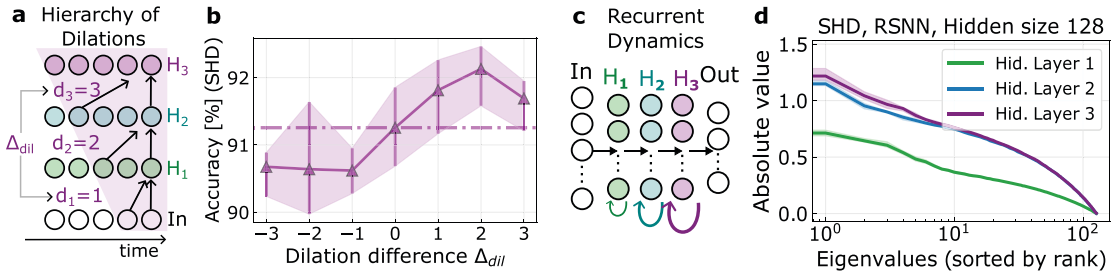

Figure 4. Temporal hierarchy in convolutional spiking neural networks. (a) Dilation in causal temporal convolutions controls the size of the receptive field. (b) Effect of the hierarchy of dilations on the SHD task. Incrementing dilation through the hidden layers improves the performance of the network. Error bars show the quartiles over 10 trials. (c) Recurrent dynamics through the layers of a recurrent SNNs (RSNN). (d) The absolute value of the eigenvalues of the recurrent weight matrix in a 3-hidden layer recurrent SNNs trained on SHD.

Results in figure 4(b) show a positive correlation between classification accuracy and temporal hierarchy in dilation, confirming the role of temporal hierarchy as an effective inductive bias even when constructed with delays. Notably, classification accuracy improves when Δ_{dil} is positive. Similarly, we can construct temporal hierarchy with the kernel size of the temporal convolution, as the kernel size is proportional to the maximum delay in a convolutional kernel: enlarging the convolutional kernel means including longer delays. Note that varying the kernel size also changes the number of trainable parameters. Once again, we show a correlation between performance and temporal hierarchy constructed with the kernel size in figure SI.8.

To further enhance the performance of SNNs, we combine kernel size and dilation hierarchies ($\Delta_{\text{ker}} = 3$, $\Delta_{\text{dil}} = 2$) and apply data augmentation. Results, summarized in table 1, demonstrate that temporal hierarchy enables SNNs to achieve competitive performance compared to state of the art. Note that 'Test as Valid' in the table indicates the use of the test set as the validation set in the SHD task. Details on the validation strategy and data augmentation are provided in Methods (section 4.3). Additional experiments on the SSC dataset using temporal convolution hierarchies also show a slight performance improvement, with results presented in table 1.

Temporal hierarchy emerging in recurrent connections Finally, we train recurrent multi-layer LIF-based SNNs without introducing any inductive bias, analyzing the optimized recurrent weight matrices (figure 4(c)). For these experiments, we use 3-hidden-layer SNNs, where all recurrent weight matrices are initialized using Xavier initialization. The memory capacity of a population of neurons is related to the eigenvalues of the recurrent weight matrix [26, 27], with higher eigenvalues corresponding to greater memory capacity at the population level. Specifically, the largest eigenvalue, known as the spectral radius, serves as a proxy for the intrinsic 'speed' of the dynamical system. After optimizing the recurrent SNNs (RSNNs) on the SHD task, we compute the eigenvalues of the recurrent weight matrices and measure their magnitudes (figure 4(d)). The results reveal a hierarchical structure across the layers, with the eigenvalue magnitudes—and in particular the spectral radius—increasing progressively through the hidden layers. We conclude that optimizing recurrent SNNs naturally leads the architecture to a temporal hierarchy.

2.4. Mathematical analysis

To illustrate the effectiveness of the temporal hierarchy in SNNs, we present a theoretical argument based on signal processing principles. Some signals follow simple patterns with single frequency components that can be detected by a single LIF neuron with an appropriately tuned time constant. For example, in figure 5(a), the purple and green spike patterns correspond to slow and fast signals, respectively, and each can be captured by a neuron with the corresponding time constant. However, more complex signals, such as speech, consist of long sequences composed of fast and short sub-patterns. A single time constant is insufficient to capture such signals effectively. This limitation is illustrated in figure 5(a), where combining the purple and green patterns results in a purple-pattern-tuned neuron also responding to the green one, leading to poor selectivity and misclassification.

This issue can be addressed through a temporal hierarchy: early layers detect short patterns and transmit this information at a reduced rate to higher layers, where longer time constants allow for gradual integration over time. Figure 5 visually explains this argument, with more detailed derivations provided in appendix SI.G.

Table 1. Performance with and without temporal hierarchy of SNNs on the spiking Heidelberg digits dataset (SHD) and the spiking speech command dataset (SSC). Note that 'CSNN' stands for Convolutional-SNN, with temporal causal convolutions acting as delayed connections. Asterisk (*) in Data Augmentation refers to a particular split of the SHD dataset referred to as 'Test as Validation', explained in Method section 4.3. Results in bold feature temporal hierarchy.

	Hierarchy	Data Augment	# hid. layers	hidden size	# params	Accuracy%
SNN (SHD)			3	32	27k	81.9 ± 1.2
	✓		3	32	27k	**85.7 ± 1.2**
			3	128	157k	84.7 ± 1.5
	✓		3	128	157k	**87.2 ± 1.6**
CSNN (SHD)			2	128	345k	91.2 ± 0.8
	✓					**92.2 ± 1.1**
	✓	✓				**94.1 ± 0.7**
	✓	✓*				**94.7 ± 0.7**
SNN (SSC)			4	128	175k	71.0 ± 1
	✓					**73.1 ± 0.3**
CSNN (SSC)		✓	2	256	1 M	78.2 ± 0.1
	✓	✓				**79.2 ± 0.2**

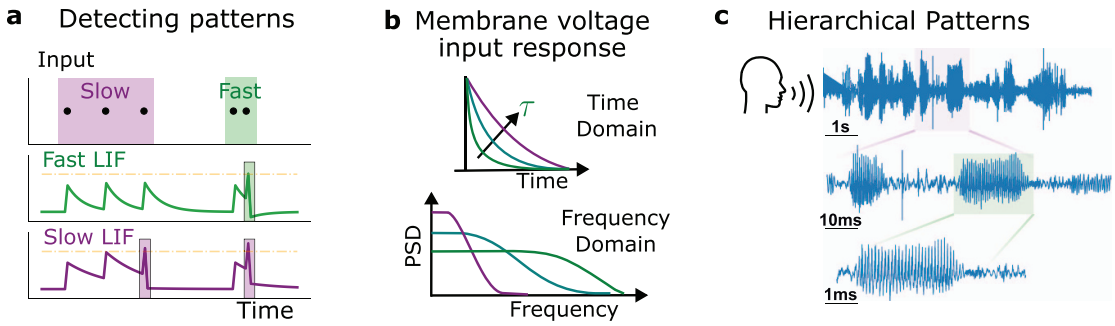

Figure 5. Temporal hierarchy is necessary to process long temporal patterns with broad frequency bands. (a) A LIF neuron with short time constants (i.e. fast LIF) can detect fast patterns (green), but not slow ones (purple). However, a LIF neuron with long time constants (slow LIF) reacts to both families of patterns and does not differentiate between them. Colored boxes indicate the outputs of the fast (green) and slow LIF neurons. (b) The impulse response of the LIF membrane with different time constants in the time domain (upper); and their corresponding filtering property in frequency domain (lower); In the time domain, the time constant indicates the temporal receptive field of the neuron, modulating the length of the temporal window to which the neuron is sensitive. In the frequency domain (power spectral density), the time constant indicates the cut-off frequency of the input signal to which the neuron is sensitive. (c) A multi-time scale signal, such as speech, contains information in a broad range of frequencies, spanning multiple temporal decades (purple/green shade highlights a slow/fast frequency component), hence requiring a hierarchy of time constants for processing.

Intuitively, a single neuron or a single layer of neurons can either respond selectively to very short temporal patterns using a short time constant or respond to both slow and fast patterns using a long time constant—but without distinguishing between them (figure 5(b)). This limitation means that a single timescale cannot effectively be selective to particular input patterns across different temporal scales.

To formalize this intuition mathematically, we model a single layer of neurons as a linear first-order low-pass filter, disregarding the non-linearity introduced by the spiking activation function in LIF neurons. We then express the membrane voltage dynamics in both the temporal and frequency domains as follows:

$$u(t^*) = \int_{-\infty}^{t^*} \mathrm{e}^{\frac{t^*-t}{\tau}} I(t)\,\mathrm{d}t, \quad \hat{u}(\omega) = \frac{1}{1+\omega\tau}\hat{I}(\omega) \tag{1}$$

where $\hat{u}(\omega)$ is the Fourier transform of the temporal signal $u(t)$, which in this case is the membrane potential of the LIF neuron in the subthreshold regime. In the time domain, a larger τ causes the membrane potential to depend on inputs from a longer history, integrating signals over an extended period. In the frequency domain, a longer τ acts as a low-pass filter with smaller cut-off frequency, reducing the neuron's sensitivity to high-frequency components and emphasizing slower signal variations.

This has implications for the time constants required to process temporal input correctly. Consider an input signal $I(t)$ which needs to be classified by integrating information in the temporal range $[0, T]$

and using frequency components in the range $[0, \Omega]$. If we assume that there needs to be a minimal difference between two classes of inputs for the SNN to perform well, then we can provide bounds for the time constants of a network with a single hidden layer (see supplementary section SI.F.2),

$$\tau_{\max} \leqslant \frac{1}{\rho\Omega}, \quad \tau_{\min} \geqslant \frac{T}{\ln(\rho^{-1})}. \tag{2}$$

where ρ is a constant that quantifies the minimum distance between two classes. Crucially, the values of T, Ω, ρ depend on the specific task to be solved, and it is possible that the two conditions cannot be satisfied for a single hidden layer SNN in particular tasks. Extending to multi-layer SNNs with a temporal hierarchy can solve this problem. In such networks, later layers integrate information over longer times, pre-processed by earlier layers which can capture high frequency components, as is required for speech processing (figure 5(c)). We can use this logic to show that such a network requires L layers to correctly solve the task:

$$L_{\text{hierarchy}} = O(\ln(T) + \ln(\Omega)) \tag{3}$$

where $L_{\text{hierarchy}}$ is the number of layers required in the case of temporal hierarchy.

In contrast, if the time constants in all hidden layers are the same (homogeneous SNN) the number of hidden layers required to integrate over the task duration T and covering all frequencies is $L_{\text{homogeneous}} = O(T\Omega)$. Importantly, within our assumptions, the number of hidden layers scales logarithmically in the case of temporal hierarchy, compared to a worst-case linear scaling $O(T\Omega)$ for homogeneous networks.

We remark that these scaling results hold within the assumptions of the analysis. In particular, the linear filter model omits the spiking nonlinearity, whose presence, together with learned weight diversity, introduces functional differentiation across layers even in homogeneous networks, partially compensating for the absence of structured temporal parameters. Furthermore, the theoretical advantage scales with the temporal bandwidth $T\Omega$ of the input, and the benchmarks used in this work, while among the best neuromorphic test cases for multi-timescale processing, do not span an extremely wide frequency range. Both factors contribute to the moderate, yet consistent, empirical gains (2%–6%) observed across our experiments.

3. Discussion

This work proposes to organize temporal computational elements in a hierarchy, from fast to slow. First, we show that organizing time constants hierarchically as an inductive bias leads to a performance improvement in common temporal benchmark tasks. Next, we optimize the time constants in SNNs via gradient descent and find that the temporal hierarchy naturally emerges. We also generalize the role of temporal hierarchy in SNNs to computational principles: it improves performance as an inductive bias in temporal causal convolutional SNNs, and it emerges through optimization in RSNNs. In all cases, we benchmark SNNs on temporal tasks (MTS-XOR, SHD, and SSC), where the evolution of the data over time encodes relevant information. As shown in our mathematical analysis, the theoretical advantage of temporal hierarchy grows with the temporal bandwidth $T\Omega$ of the input signal, and the logarithmic depth scaling holds under a linear filter approximation. In practical settings in SNNs, where spiking nonlinearities and learned weights provide additional functional differentiation across layers, and where available benchmarks do not span an extremely wide frequency range, the empirical gains are more moderate but remain consistent (2%–6% accuracy improvement). The benefits of temporal hierarchy might be even greater when applied to tasks with more complex temporal structures.

Implications for neuromorphic computing The field of neuromorphic engineering aims to build efficient hardware platforms for SNNs. Many existing neuromorphic chips [28–30] support the LIF neuron model, and some circuits to implement dendritic delays have been developed [10, 31, 32]. This makes temporal hierarchy a valuable feature that most neuromorphic hardware designers can easily leverage. For example, in figure 2(c), we show that temporal hierarchy allows reducing the size of an SNN by 5×, while maintaining the same performance on the SHD task. In this manner, the energy consumption and memory footprint of a neuromorphic chip running an SNN can be considerably reduced. Furthermore, temporal hierarchy has been demonstrated on multiple temporal processing primitives. This indicates

that temporal hierarchy can be customized to the specifics of each neuromorphic chip implementation, providing a performance benefit independent of the chip's design specifications.

Temporal hierarchy and heterogeneity Our work formalizes temporal hierarchy as a structured across-layer progression of temporal processing speeds. A natural question is whether this fast-to-slow direction is a general computational principle or merely one of several viable configurations. Several observations support its generality. First, the neuroscience literature documents this progression across primate cortical areas [1–3], suggesting it is not task-specific. Second, our emergence-from-optimization experiments (figures 3 and 4(d)) show that gradient descent independently rediscovers this direction across different temporal parameters. Third, recent work [33] reports that optimized axonal delays also grow progressively in deeper layers, providing independent corroboration through yet another temporal parameter. Importantly, our experiments with negative $\Delta\tau$ (figures 2(a) and (d)) show that the reversed arrangement (slow-to-fast) consistently degrades performance below the homogeneous baseline, indicating that weight learning cannot fully compensate for a misaligned temporal structure. Our mathematical analysis provides a principled explanation: early layers must preserve high-frequency information (equation (2), upper bound on τ), while deeper layers must integrate over the full task duration (equation (2), lower bound on τ), making the fast-to-slow direction the only arrangement that satisfies both constraints simultaneously. It is worth noting that the concept of temporal hierarchy is distinct from, and complementary to, temporal heterogeneity within a single layer, as explored in [5, 6]. Recent work on dual memory pathways [34] demonstrates that co-locating fast and slow dynamics within the same layer can also improve temporal processing. While within-layer heterogeneity enriches the temporal feature space of individual layers, across-layer hierarchy provides the structured progression that reduces the required network depth from $\mathcal{O}(T\Omega)$ to $\mathcal{O}(log(T\Omega))$ (section 2.4). Combining both principles is a promising direction for future work.

Scaling temporal hierarchy Beyond the SNN domain, the principle of hierarchical temporal processing has been adopted independently in the recurrent neural network literature. The Clockwork RNN [35] partitions hidden units into modules operating at different clock rates, the Hierarchical Multiscale RNN [36] learns layer-wise temporal resolutions, and the recent Hierarchically Gated RNN [37] enforces monotonically increasing forget gate lower bounds through depth, with gating factors playing a role analogous to the time constants studied here. This convergence across neuroscience-inspired and mainstream deep learning communities suggests that temporal hierarchy is a general computational principle whose benefits are likely to persist at larger scales.

Outlook In addition to our demonstration of the benefit of temporal hierarchy in multiple settings, there are numerous ways to extend this work. On the one hand, more complex neuronal dynamics can be leveraged to form temporal hierarchies. The true potential of the dynamics of, for example, Izhikevich neurons [38] or multi-compartment models could be unlocked by a hierarchical setting of their parameters. For example, one could build a temporal hierarchy exploiting complex computational primitives of biologically plausible neuron models, such as plateau potentials, spike bursts, and intrinsic plasticity. On the other hand, the benefit of temporal hierarchy could be even greater when tested on datasets with a higher degree of temporal information. In some cases, temporal hierarchy is a natural feature of data. In speech, information is represented across multiple levels: phonemes, words, phrases, and semantics. We expect a single model that tackles all these input representations simultaneously to benefit greatly from a temporal hierarchy.

4. Methods

4.1. SNN implementation

For all our experiments, we implement SNNs using the JAX [39] Python library to benefit from auto-differentiation and speed of execution. We run the experiments on two A6000 GPUs, using a system with an AMD Ryzen Threadripper CPU and 128 GB of RAM. SNNs are formalized based on the LIF equation with soft reset:

$$\tau \dot{u}(t) = -u(t) + f(x(t), W) - u_{\text{th}} y(t)$$
$$y(t) = \theta(u(t) - u_{\text{th}})$$

where, $u(t) \in \mathcal{R}^N$ is the membrane voltage of the neurons, $\tau \in \mathcal{R}^N$ their time constants, $W \in \mathcal{R}^{M,N}$ or $\mathcal{R}^{K,M,N}$ the input weights and $x(t) \in \mathcal{R}^M$ the layer's inputs. N is number of neurons, M number of inputs, K kernel size (when in convolution mode). The neuron emits a spike ($y(t) \in \mathcal{R}^N$) whenever

the membrane voltage $u(t)$ crosses the threshold u_{th}. We set $u_{\text{th}} = 1$ for all our experiments. After a spike, the membrane voltage is reset: we chose a 'soft reset' mechanism, i.e. the membrane voltage is not immediately reset to zero, but it is decreased proportionally to the threshold voltage.

As it is well known, the non-linearity of a LIF neuron (Heaviside function) is non-differentiable. To allow the gradient to flow, we use the Surrogate Gradient technique [8], replacing the derivative of the Heaviside function with a 'box function', as in [40].

Based on the equations above, we build layers of spiking neurons linked either via all-to-all feed-forward connectivity

$$f(I(t), W) = x(t)\, W \quad \text{with} \quad W \in \mathcal{R}^{M,N}$$

or by 1D temporal causal convolution

$$f(I(t), W) = x(t) * W \quad \text{with} \quad W \in \mathcal{R}^{K,M,N}$$

In the latter case, we pad the input sequence $I(t)$ with as many zeros as $K-1$ to make the temporal convolution causal.

A normalization layer processes the result of such linear operation. For all our experiments we use 'temporal' batchnorm, a technique introduced in [40]. The conventional batchnorm operation is applied not only to the batch dimension but also to the temporal dimension. In all experiments, the normalization statistics (running average and standard deviation) are neither tracked nor updated during training, simple batch statistics are used instead.

In each layer l, neurons are initialized with a certain time constant $\tau(l)$. We draw these time constants from a Uniform distribution with a given mean and a width of 20% of the mean. Note that temporal hierarchy is formed by controlling the mean of the time-constant distribution across the layers. Multiple such layers are concatenated to form either a multi-layer spiking perceptron (experiments in figure 2) or a convolutional SNN (figure 4). The output layer, instead, always features Leaky-Integrators neurons, i.e. non-spiking neurons. The time constant for such output neurons is 200 ms for all experiments. The values of other relevant hyperparameters can be found in supplementary table 2.

Loss function All experiments involve supervised classification tasks, for which cross-entropy is chosen. The output logits are computed as in [40], thus computing the softmax of the output neurons membrane voltage at every time step, integrating these values over the whole duration of the task, and applying the softmax once again on the obtained values. The resulting loss function averages the cross-entropy over the task duration:

$$\mathcal{L}_{\text{cumsum}} = -\frac{1}{B}\sum_{i}^{B} \log\left(\frac{\exp\left(\text{out}\left[i, y_i\right]\right)}{\sum_{n}^{N}\exp\left(\text{out}\left[i, n\right]\right)}\right) \quad \text{with} \quad \text{out} = \sum_{t}^{T}\text{softmax}\left(u\left[t\right]\right)$$

where B is the batch size, N number of output neurons, y the desired label, $\text{out} \in \mathcal{R}^{B,N}$ the output neurons membrane voltage passed through the softmax function and summed over the task duration T:

In the case of the MTS-XOR task, we opted for a 'max-over-time' loss function, typical in many SNNs in the literature [8, 19]. This loss function looks for the maximum value for the membrane voltage of the output neuron over a certain period. In the case of the MTS-XOR, this period corresponds to the presentation of the cues in Channel B. The 'max-over-time' loss function equation is the following:

$$\mathcal{L}_{\text{max}} = -\frac{1}{B}\sum_{i}^{B} \log\left(\frac{\exp\left(\max_T\left(\text{u}_\text{L}\left[i, y\right]\right)\right)}{\sum_{n}^{N}\exp\left(\max_T\left(\text{u}_\text{L}\left[i, n\right]\right)\right)}\right)$$

where $\max_T$ indicates the selection of the maximum value of the membrane voltage at the output layer u_L over the selection period T.

Regularization Dropout with a probability of 10% is applied in the experiments in figure 2 while the probability is increased at 40% in figure 4. L2-regularization with a magnitude of 10^{-4} is utilized for experiments with the convolutional SNNs. When training the time constants of neurons (figure 2), a regularization term is applied to prevent unreasonable values, i.e. negative values and too large values leading to numerical instability. The resulting regularization term on the time constant is a re-scaled L2-norm:

$$\mathcal{L}_{\tau,L2} = \sum_{l}^{L} L2\left(\tau\left(l\right) - \text{mean}\left[\tau\left(l\right)\right]\right)$$

where $\tau(l)$ is the time constant at a given hidden layer, with L hidden layers in total.

Table 2. table of the hyperparameters.

		Batch Size	Epochs	Dropout	LR, decay [epochs start, decay]	Initialization	Loss	τ_μ [s]	τ_{out} [s]
MTS-XOR	FF	512	60	0.1	0.01, linear [25, 0.5]	Xavier	$\mathcal{L}_{\text{max}}$	0.3	0.2
SHD/SSC	FF	256	60	0.1	0.01, linear [25, 0.5]	Xavier	$\mathcal{L}_{\text{cum}}$	0.1	0.2
	Conv	256	100	0.4	0.01, linear [25, 0.5]	Xavier	$\mathcal{L}_{\text{cum}}$	0.1	0.2

4.2. Hyperparameters

We list in table 2 the hyperparameters used in the experiments. The acronyms and symbols refer to: learning rate (LR), feed-forward (FF), convolutional (conv), max-over-time loss function ($\mathcal{L}_{\text{max}}$), sum-of-softmax-output ($\mathcal{L}_{\text{cum}}$), output neuron time constant τ_{out}, mean hidden layer time constant τ_μ.

4.3. Dataset composition and augmentation

In all experiments, the datasets feature training, validation, and testing subsets. However, the SHD was not originally split in this way; it only features a training and testing set. As in [10, 41], we subdivide the training set into two parts—with 80% and 20% proportion—of which the smaller portion is the validation set. While we train on the resulting training set only, hyperparameters are optimized on the validation set. In the literature, it is, however, common to perform the SHD benchmark task with a different methodology, which we refer to as 'Test as Valid' in table 1. The test set is utilized as the validation set, and performance (along with hyper-parameter optimization and validation early-stopping) is evaluated on the test set. As this method leaks information from the test set into the training phase, the generalization capabilities of the model can no longer be evaluated properly. With the sole scope of comparing performance with works utilizing such validation scheme [12, 42], we also report results obtained with this method ('Test as Valid'). A thorough comparison with the state-of-the-art in SHD is reported in the Supplementary Information (table SI.1). In some experiments on the convolutional SNN section (table 1) we utilize data augmentation to improve the generalization performance of the model. In particular, the only transformation we apply is a transposition of the input channels, equivalent to a frequency shift of the speaker's voice. For each data point, we draw a number with a mean of 10 and a standard deviation of 5, representing the frequency shift of the data. We then pad the input sequence with as many channels as the drawn frequency shift and truncate the exceeding channels to recover the original dataset input size (700 channels). The frequency shift can be positive or negative with 0.5 probability. Each sample in a batch of data has an independent frequency shift.

Acknowledgments and funding

We want to thank the EIS-Lab, especially Tristan Torchet, for assisting with the code implementation. The presented work has received funding from the Swiss National Science Foundation Starting Grant Project UNITE (TMSGI2-211 461).

Data availability statement

The Spiking Heidelberg Datasets [19] and Spiking Speech Command [21] datasets are publicly accessible. All other measured data are freely available upon request.

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/EIS-Hub/hSNN [43].

Additional results and theory available at https://doi.org/10.1088/2634-4386/ae7ab5/data1.

Code availability

All software programs used in the presentation of the article are freely available upon publication of the article.

Author contributions

Filippo Moro 0000-0002-5279-6309
Conceptualization (lead), Data curation (lead), Investigation (lead), Methodology (lead), Project administration (lead), Software (lead), Validation (lead), Visualization (lead), Writing – original draft (lead), Writing – review & editing (lead)

Pau Vilimelis Aceituno 0000-0002-1218-4009
Conceptualization (supporting), Formal analysis (lead), Methodology (equal), Writing – original draft (supporting), Writing – review & editing (equal)

Laura Kriener 0000-0001-5275-9199
Investigation (supporting), Writing – original draft (supporting), Writing – review & editing (supporting)

Melika Payvand 0000-0001-5400-067X
Conceptualization (supporting), Funding acquisition (lead), Project administration (equal), Supervision (equal), Visualization (supporting), Writing – original draft (equal), Writing – review & editing (equal)

References

[1] Murray J D *et al* 2014 A hierarchy of intrinsic timescales across primate cortex *Nat. Neurosci.* **17** 1661–3
[2] Senkowski D and Engel A K 2024 Multi-timescale neural dynamics for multisensory integration *Nat. Rev. Neurosci.* **25** 625–42
[3] Gao R, Van den Brink R L, Pfeffer T and Voytek B 2020 Neuronal timescales are functionally dynamic and shaped by cortical microarchitecture *elife* **9** e61277
[4] Zeraati R, Levina A, Macke J H and Gao R 2024 Neural timescales from a computational perspective (arXiv:2409.02684)
[5] Perez-Nieves N, Leung V C H, Dragotti P L and Goodman D F M 2021 Neural heterogeneity promotes robust learning *Nat. Commun.* **12** 5791
[6] Zheng H, Zheng Z, Hu R, Xiao B, Wu Y, Yu F, Liu X, Li G and Deng L 2024 Temporal dendritic heterogeneity incorporated with spiking neural networks for learning multi-timescale dynamics *Nat. Commun.* **15** 277
[7] Indiveri G and Sandamirskaya Y 2019 The importance of space and time for signal processing in neuromorphic agents: The challenge of developing low-power, autonomous agents that interact with the environment *IEEE Signal Process. Mag.* **36** 16–28
[8] Neftci E O, Mostafa H and Zenke F 2019 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
[9] Yin B, Corradi F and Bohté S M 2023 Accurate online training of dynamical spiking neural networks through forward propagation through time *Nat. Mach. Intell.* **5** 518–27
[10] D'Agostino S, Moro F, Torchet T, Demirağ Y, Grenouillet L, Castellani N, Indiveri G, Vianello E and Payvand M 2024 Denram: neuromorphic dendritic architecture with rram for efficient temporal processing with delays *Nat. Commun.* **15** 3446
[11] Göltz J, Weber J, Kriener L, Lake P, Payvand M and Petrovici M A 2024 DelGrad: exact event-based gradients for training delays and weights on spiking neuromorphic hardware *Nat. Comm.* **16** 8245
[12] Hammouamri I, Khalfaoui-Hassani I and Masquelier T 2024 Learning delays in spiking neural networks using dilated convolutions with learnable spacings *Int. Conf. on Learning Representations* vol 2024 pp 17890–903 (available at: https://proceedings.iclr.cc/)
[13] Graves A 2013 Generating sequences with recurrent neural networks (arXiv:1308.0850)
[14] Khajehabdollahi S, Zeraati R, Giannakakis E, Schäfer T J, Martius G and Levina A 2024 Emergent mechanisms for long timescales depend on training curriculum and affect performance in memory tasks *Int. Conf. on Learning Representations* pp 19455–82 (available at: https://proceedings.iclr.cc/)
[15] Gerstner W, Kistler W M, Naud R and Paninski L 2014 *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition* (Cambridge University Press)
[16] Masquelier T 2018 Stdp allows close-to-optimal spatiotemporal spike pattern detection by single coincidence detector neurons *Neuroscience* **389** 133–40
[17] Moro F *et al* 2022 Neuromorphic object localization using resistive memories and ultrasonic transducers *Nat. Commun.* **13** 3506
[18] Göltz J *et al* 2021 Fast and energy-efficient neuromorphic deep learning with first-spike times *Nat. Mach. Intell.* **3** 823–35
[19] Cramer B, Stradmann Y, Schemmel J and Zenke F 2020 The heidelberg spiking data sets for the systematic evaluation of spiking neural networks *IEEE Trans. Neural Netw. Learn. Syst.* **33** 2744–57
[20] Weidel P and Sheik S 2021 Wavesense: efficient temporal convolutions with spiking neural networks for keyword spotting (arXiv:2111.01456)
[21] Warden P 2018 Speech commands: a dataset for limited-vocabulary speech recognition (arXiv:1804.03209)
[22] Le Q V, Jaitly N and Hinton G E 2015 A simple way to initialize recurrent networks of rectified linear units (arXiv:1504.00941)
[23] Sun P, Eqlimi E, Chua Y, Devos P and Botteldooren D 2023 Adaptive axonal delays in feedforward spiking neural networks for accurate spoken word recognition *ICASSP 2023 - 2023 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* pp 1–5
[24] Sun P, Zhu L and Botteldooren D 2022 Axonal delay as a short-term memory for feed forward deep spiking neural networks *ICASSP 2022 - 2022 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* pp 8932–6
[25] Deckers L, Van Damme L, Van Leekwijck W, Tsang I J and Latré S 2024 Co-learning synaptic delays, weights and adaptation in spiking neural networks *Front. Neurosci.* **18** 1360300
[26] Aceituno P V, Yan G and Liu Y-Y 2020 Tailoring echo state networks for optimal learning *Iscience* **23** 101440
[27] Farkaš I, Bosák R and Gergel' P 2016 Computational analysis of memory capacity in echo state networks *Neural Netw.* **83** 109–20

[28] Pehle C, Billaudelle S, Cramer B, Kaiser J, Schreiber K, Stradmann Y, Weis J, Leibfried A, Müller E and Schemmel J 2022 The brainscales-2 accelerated neuromorphic system with hybrid plasticity *Front. Neurosci.* **16** 795876
[29] Richter O, Wu C, Whatley A M, Köstinger G, Nielsen C, Qiao N and Indiveri G 2024 Dynap-se2: a scalable multi-core dynamic neuromorphic asynchronous spiking neural network processor *Neuromorph. Comput. Eng.* **4** 014003
[30] Orchard G, Frady E P, Rubin D B D, Sanborn S, Shrestha S B, Sommer F T and Davies M 2021 Efficient neuromorphic signal processing with loihi 2 *2021 IEEE Workshop on Signal Processing Systems (SiPS)* (IEEE) pp 254–9
[31] Ramakrishnan S, Wunderlich R, Hasler J and George S 2013 Neuron array with plastic synapses and programmable dendrites *IEEE Trans. Biomed. Circuits Syst.* **7** 631–42
[32] Wang Y and Liu S-C 2010 Multilayer processing of spatiotemporal spike patterns in a neuron with active dendrites *Neural Comput.* **22** 2086–112
[33] Sun P, Achterberg J, Su Z, Goodman D F M and Akarca D 2025 Exploiting heterogeneous delays for efficient computation in low-bit neural networks (arXiv:2510.27434)
[34] Sun P, Su Z, Achterberg J, Indiveri G, Goodman D F M and Akarca D 2025 Algorithm-hardware co-design of neuromorphic networks with dual memory pathways (arXiv:2512.07602)
[35] Koutnik J, Greff K, Gomez F and Schmidhuber J 2014 A clockwork rnn *Int. Conf. on Machine Learning* (PMLR) pp 1863–71
[36] Chung J, Ahn S and Bengio Y 2016 Hierarchical multiscale recurrent neural networks (arXiv:1609.01704)
[37] Qin Z, Yang S and Zhong Y 2023 Hierarchically gated recurrent neural network for sequence modeling *Advances in Neural Information Processing Systems* vol 36 pp 33202–21
[38] Izhikevich E M 2001 Resonate-and-fire neurons *Neural Netw.* **14** 883–94
[39] Bradbury J, Frostig R, Hawkins P, Johnson M J, Leary C, Maclaurin D, Necula G, Paszke A, VanderPlas J, Wanderman-Milne S and Zhang Q JAX: composable transformations of Python+NumPy programs 2018 (available at: http://github.com/google/jax)
[40] Bittar A and Garner P N 2022 A surrogate gradient spiking baseline for speech command recognition *Front. Neurosci.* **16** 865897
[41] Nowotny T, Turner J P and Knight J C 2025 Loss shaping enhances exact gradient learning with eventprop in spiking neural networks *Neuromorph. Comput. Eng.* **5** 014001
[42] Schöne M, Sushma N M, Zhuge J, Mayr C, Subramoney A and Kappel D 2024 Scalable event-by-event processing of neuromorphic sensory signals with deep state-space models *2024 Int. Conf. on Neuromorphic Systems (ICONS)* (*Arlington, VA, USA*) pp 124–31
[43] Moro F 2025 *Temporal Hierarchy in Spiking Neural Networks* (available at: https://github.com/EIS-Hub/hSNN)