Spike frequency adaptation supports network computations on temporally dispersed information

Short title: Temporal computing with spike frequency adaptation

Darjan Salaj[1,†], Anand Subramoney[1,†], Ceca Kraišniković[1,†], Guillaume Bellec[1,2], Robert Legenstein[1], Wolfgang Maass[1,*]

December 4, 2020

[1]*Institute of Theoretical Computer Science, Graz University of Technology, Inffeldgasse 16b, Graz, Austria*
[2]*Laboratory of Computational Neuroscience, Ecole Polytechnique Fédérale de Lausanne (EPFL), Bâtiment AAB, offices 135-141, CH-1015 Lausanne*
[†] *These authors contributed equally to this work.*
[*] *To whom correspondence should be addressed; E-mail: maass@igi.tugraz.at.*

Abstract

For solving tasks such as recognizing a song, answering a question, or inverting a sequence of symbols, cortical microcircuits need to integrate and manipulate information that was dispersed over time during the preceding seconds. Creating biologically realistic models for the underlying computations, especially with spiking neurons and for behaviorally relevant integration time spans, is notoriously difficult. We examine the role of spike frequency adaptation in such computations and find that it has a surprisingly large impact. The inclusion of this well known property of a substantial fraction of neurons in the neocortex — especially in higher areas of the human neocortex — moves the performance of spiking neural network models for computations on network inputs that are temporally dispersed from a fairly low level up to the performance level of the human brain.

Introduction

Since brains have to operate in dynamically changing environments, neural networks of the brain need to be able to solve "temporal computing tasks" that require integration and manipulation of temporally dispersed information from continuous input streams on the behavioral time scale of seconds. Models for neural networks of the brain have inherent difficulties in modeling such temporal computations on the time scale of seconds since spikes and postsynaptic potentials take place on the much shorter time scales of milliseconds and tens of milliseconds. Most work on models for temporal computing in neural networks of the brain has focused on networks of non-spiking neurons and temporal computing tasks where one or a few bits have to be stored in working memory without requiring computational operations other than storage and recall. We address the question of how biologically more realistic spike-based neural network models can solve a range of generic temporal computing tasks that require extraction of a fairly large amount of temporally dispersed information from continuous input streams. These tasks also require information stored in working memory to be continuously updated, as required, for example, for speech understanding or manipulations of sequences of symbols. So far, these tasks could not be solved by spiking neural network models.

We examine the impact of spike frequency adaptation (SFA) of neurons on temporal computing capabilities of spiking neural networks (SNNs). SFA denotes a feature of spiking neurons where their preceding firing activity transiently increases their firing threshold. Experimental data from

the Allen Institute (Allen Institute, 2018) show that a substantial fraction of excitatory neurons of the neocortex, ranging from 20% in mouse visual cortex to 40% in the human frontal lobe, exhibit SFA, see Fig. 1A for sample responses of neurons with different levels of SFA. Although a rigorous survey of time constants of SFA is still missing, the available experimental data show that SFA does produce history dependence of neural firing on the time scale of seconds, in fact, up to 20 seconds according to (Pozzorini et al., 2013, 2015). The biophysical mechanisms behind SFA include inactivation of depolarizing currents and the activity-dependent activation of slow hyperpolarizing or shunting currents (Gutkin and Zeldenrust, 2014; Benda and Herz, 2003). These have already been implicated in cellular short-term memory (Marder et al., 1996; Turrigiano et al., 1996). SFA has also been argued to contribute to a number of other important features of brain networks (Gutkin and Zeldenrust, 2014). On the single neuron level, these features include the enhancement of sensitivity to synchronous input and effects on the frequency response curve (Benda et al., 2010; Ermentrout, 1998; Wang, 1998). On the network level, SFA may influence population coding or benefit Bayesian inference (Kilpatrick and Ermentrout, 2011; Deneve, 2008). The contribution of SFA to temporal computing capabilities of recurrent SNNs had first been examined in (Bellec et al., 2018). The role of SFA for language processing in feedforward networks was subsequently examined in (Fitz et al., 2020).

We investigate here the role of SFA for a range of demanding temporal computing tasks, including the well-known 12AX task and manipulations of sequences of symbols that are also used for testing cognitive capabilities of brains. SFA contributes surprisingly strongly to temporal computing, in spite of the counter-intuitive fact that SFA reduces — rather than enhances — persistent firing. We introduce the negative imprinting principle as an explanation for this. Finally, we discuss the neural codes that emerge in neural networks with SFA for manipulations on sequences of symbols, and compare them with neural codes for corresponding tasks in the brain (Barone and Joseph, 1989; Liu et al., 2019; Carpenter et al., 2018).

Results

Network model

We employ a very simple model for SFA, the generalized leaky integrate-and-fire model $GLIF_2$ from (Teeter et al., 2018; Allen Institute, 2017). A practical advantage of this simple model is that it can be very efficiently simulated and is amenable to gradient descent training methods. It assumes that the firing threshold $A(t)$ of a leaky integrate-and-fire (LIF) neuron contains a variable component $a(t)$ that increases by a fixed amount after each of its spikes $z(t)$ (Fig. 1B), and then decays exponentially back to 0. This variable threshold models the inactivation of voltage-dependent sodium channels in a qualitative manner. We write $z_j(t)$ for the spike output of neuron j, that switches from 0 to 1 at time t when the neuron fires at time t, and otherwise has value 0. With this notation one can define the SFA model by the equations:

$$\begin{aligned} A_j(t) &= v_{\text{th}} + \beta a_j(t), \\ a_j(t+1) &= \rho_j a_j(t) + (1-\rho_j) z_j(t), \end{aligned} \tag{1}$$

where v_{th} is the constant baseline of the firing threshold $A_j(t)$, and $\beta > 0$ scales the amplitude of the activity-dependent component. The parameter $\rho_j = \exp\left(\frac{-1}{\tau_{a,j}}\right)$ controls the speed by which $a_j(t)$ decays back to 0, where $\tau_{a,j}$ is the adaptation time constant of neuron j. Here, we used a unitary time step for simplicity, see Methods for equations with arbitrary discrete time steps δt (which we chose to be 1 ms) and further details on neuron and synapse models. We consider in this article recurrent networks of LIF neurons where some fraction of those is equipped with SFA. Neurons in the recurrent network project to readout neurons which produce the output of the network, see Fig. 1C.

In order to demonstrate the contribution of SFA and other mechanisms on temporal computing capabilities of SNNs, we optimized the weights of the SNN for each of a variety of generic temporal computing tasks. We used Backpropagation through time (BPTT) for this, which is arguably the best performing optimization method for SNNs that is currently known. The non-differentiability

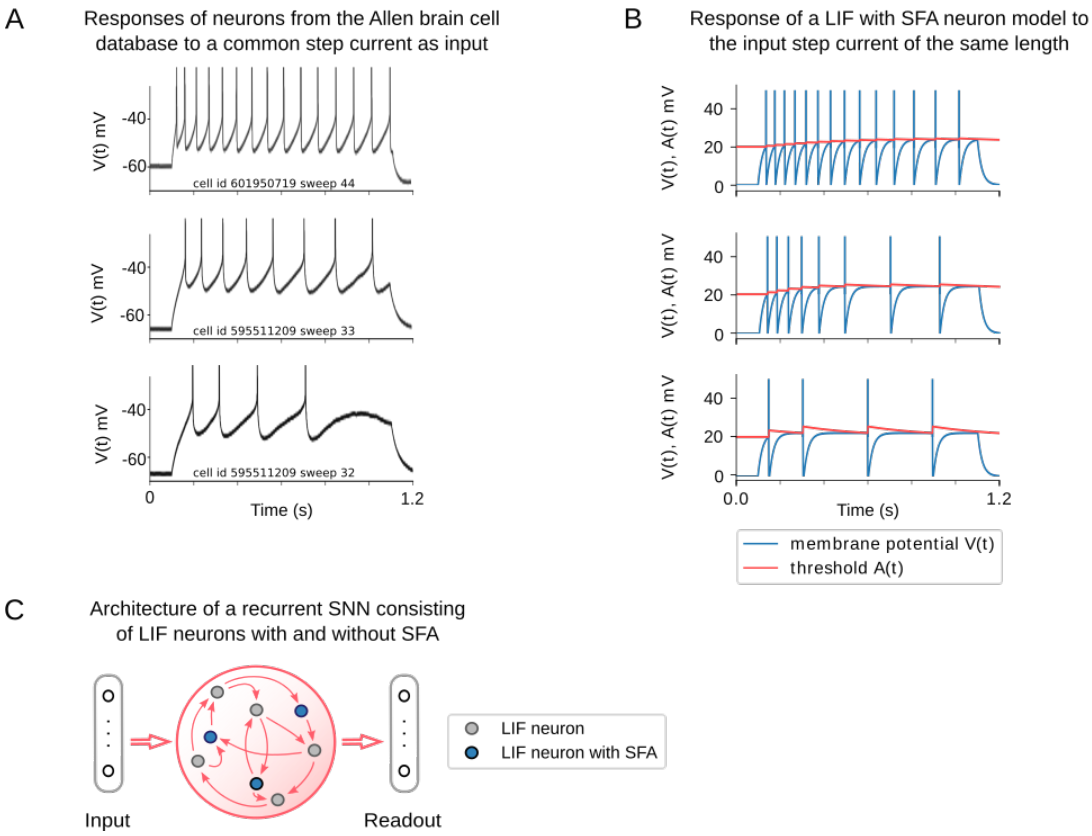

Figure 1: Experimental data on neurons with SFA, and a simple model for SFA. (**A**) The response to a 1-second long step current is displayed for three sample neurons from the Allen brain cell database (Allen Institute, 2018). The cell id and sweep number identify the exact cell recording in the Allen brain cell database. (**B**) The response of a simple LIF neuron model with SFA to the 1-second long step current. Neuron parameters used: top row $\beta = 0.5\,\mathrm{mV}$, $\tau_a = 1\,\mathrm{s}$, $I_{input} = 0.024\,\mathrm{A}$; middle row $\beta = 1\,\mathrm{mV}$, $\tau_a = 1\,\mathrm{s}$, $I_{input} = 0.024\,\mathrm{A}$; bottom row $\beta = 1\,\mathrm{mV}$, $\tau_a = 300\,\mathrm{ms}$, $I_{input} = 0.022\,\mathrm{A}$. (**C**) Symbolic architecture of recurrent SNN consisting of LIF neurons with SFA.

of spiking neuron models can be overcome quite well with the help of a suitable pseudo-derivative (Bellec et al., 2018).

SFA provides working memory simultaneously for many pieces of information, and yields powerful generalization capability

In order to elucidate the mechanisms by which SFA supports temporal computing capabilities of SNNs we first consider simpler tasks where the stored information does not have to be updated. Simultaneously we examine whether the working memory contribution of SFA scales up to working memory demands larger than in the commonly considered task where just a single bit has to be stored in working memory. Obviously, brains have a much larger working memory capacity insofar as they are able to store many salient bits of information from a previously seen image or movie or text. Furthermore, they are able to ignore irrelevant parts of complex sensory input streams. Some of these demanding aspects are captured in the task that is considered in Fig. 2. This task aims at capturing the need to remember a fair number of higher-level features that are needed for later recall of the content of an image in a higher visual area such as area IT (inferior temporal cortex). There, a 20-dimensional input stream of bits provides a continuous stream of input patterns, where each pattern can be visualized as 4×5 image, as indicated in the top row of Fig. 2. Occasionally, a pattern in the input stream is marked as being salient through simultaneous activation of a STORE command in a separate input channel, corresponding for example to an attentional signal from a

higher brain area. The task is to reproduce during a RECALL command the pattern that had been presented during the most recent STORE command. Delays between STORE and RECALL ranged from 200 to 1600 ms. 20 binary values were simultaneously extracted as network outputs during RECALL by rounding the output values of 20 linear readout neurons. We found that an SNN consisting of 500 neurons with SFA, whose adaptive firing threshold had a time constant of $\tau_a = 800$ ms, was able to solve this task with an accuracy above 99%. SFA was essential for this behavior, because the recall performance of a recurrent network of LIF neurons without SFA, trained in exactly the same way, stayed at chance level (see Methods).

Generalization of SFA-enhanced temporal computations to unseen inputs. Humans can retain previously unseen stimuli in short-term memory. In order to probe whether SNNs with SFA are capable of such generalization, we made sure that none of the patterns shown during testing had occurred during training, and in fact, had a Hamming distance of at least 5 bits to all training patterns. The resulting recall performance of the SNN with SFA was 99.09%, i.e., 99.09% of the stored feature vectors were perfectly reproduced during recall. Note that in contrast to most models for working memory, we require that this SNN is able to store content other than what was used when the values of its synaptic weights were determined. A sample segment of a test trial is shown in Fig. 2, with the activity of input neurons at the top and the activation of readout neurons at the bottom.

Negative imprinting principle. Fig. 2 elucidates how neurons with SFA support temporal computing: The 3rd to last row shows the temporal dynamics of 25 selected neurons with SFA. One fraction of these neurons, with firing thresholds drawn in blue (second to last row), fires strongly during a STORE command. A complementary set of neurons with SFA, with trajectories of firing thresholds drawn in red, fires during the subsequent RECALL command. We propose that the non-firing of the neurons with thresholds drawn in blue during the RECALL command signals to the readout neurons which of the 20 bits were active during STORE, and thereby enables them to reproduce this 20-bit pattern. We refer to this coding method as negative imprinting principle. It is of general interest insofar as one often focuses on the information that is transmitted by spikes, and forget that non-spiking can also transmit information — in fact in a much more energy-efficient manner.

Interestingly, the firing activity of the network was rather low during the delay between STORE and RECALL. Furthermore, a powerful classifier — a Support Vector Machine (SVM) trained on the network activity during the delay — was not able to decode the stored feature vector from the firing activity (the decoding accuracy during the delay was 4.38%, as opposed to 100% decoding accuracy during RECALL; see Methods). Hence the type of working memory that the SNN with SFA exhibits during the STORE-RECALL task is related to the activity-silent form of working memory in the human brain that had been examined in the experiments of (Wolff et al., 2017).

If the content of working memory is encoded by an attractor of the network dynamics, one would expect that the neural code for the content of the working memory changes little between encoding and a subsequent network reactivation. In contrast to this principle, it had been shown for the human brain in (Wolff et al., 2017) that the representation of working memory content changes significantly between memory encoding and subsequent network reactivation during the delay by an "impulse stimulus": A classifier trained on the network activity during encoding was not able to classify the memory content during a network reactivation in the delay, and vice versa. This experimental result from the human brain is consistent with the negative imprinting principle. We also tested directly whether such change of neural codes occurs in our model for the STORE-RECALL task. We found that a classifier trained for decoding the content of working memory during STORE was not able to decode this content during RECALL, and vice versa (see Methods). Hence our model is in this regard consistent with the experimental data of (Wolff et al., 2017).

No precise alignment between time constants of SFA and working memory duration is needed. Experimental data from the Allen Institute database suggest that different neurons exhibit a diversity of SFA properties. We show that correspondingly a diversity of time constants of SFA in different neurons provides high performance for temporal computing. We consider for simplicity the one-dimensional version of the task of Fig. 2, where just a single bit needs to be

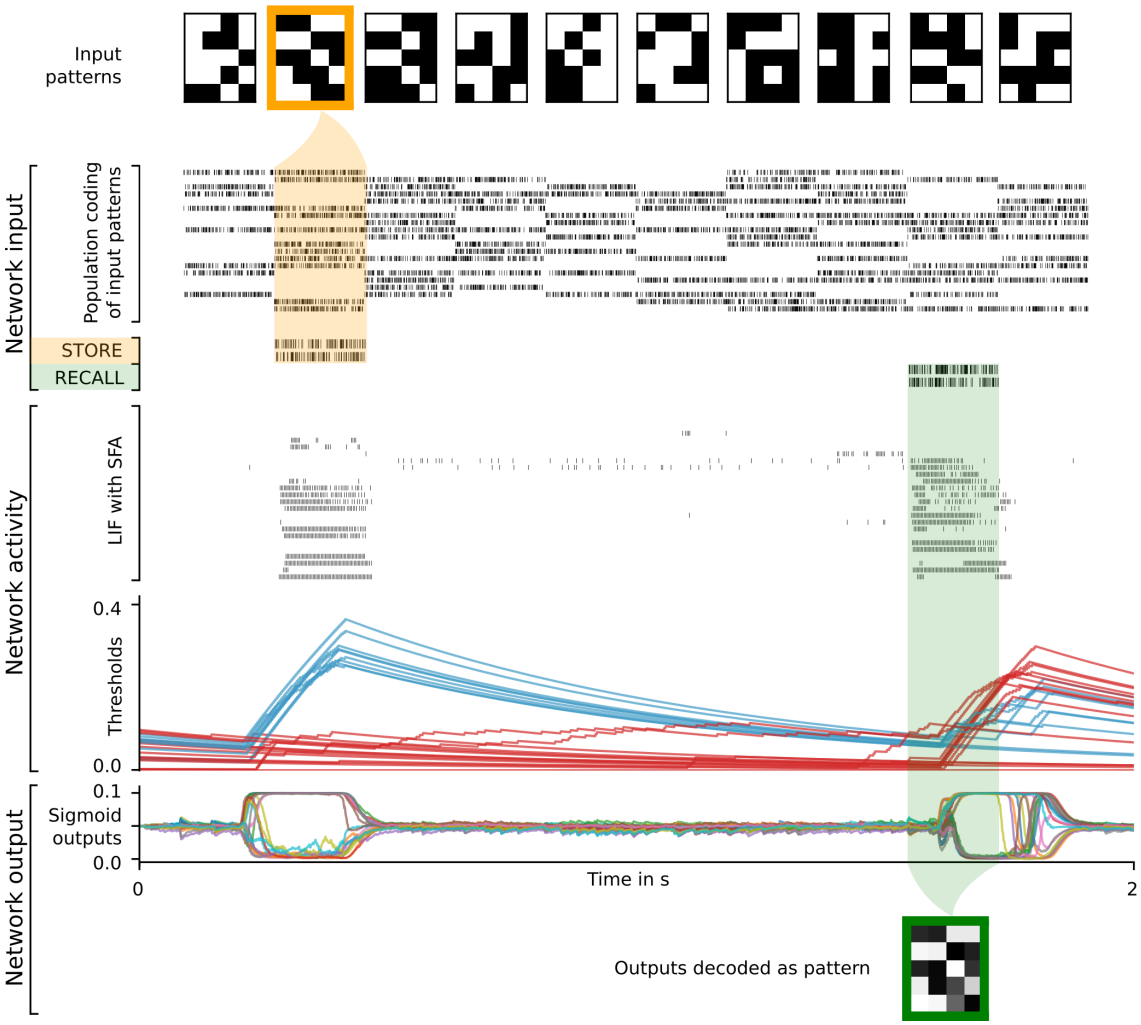

Figure 2: Sample trial of 20-dimensional STORE-RECALL task. Rows top to bottom: Stream of randomly drawn 20-dimensional input patterns, represented by the firing activity of 20 populations of input neurons (a subsample is shown), firing activity of two additional populations of input neurons for the STORE and RECALL commands, firing activity of 25 sample LIF neurons with SFA in the SNN (we first ordered all neurons with regard to the variance of their dynamic firing thresholds, and then picked every 20th), the temporal evolution of the firing thresholds of these 25 neurons, traces of the activation of 20 sigmoidal readout neurons, and their average value during the 200 ms time window of the RECALL command represented by grey values. During the RECALL command (green shading) the network successfully reproduced the pattern that had been given as input during the preceding STORE command (yellow shading). Coloring of the threshold traces in blue or red was done after visual inspection to highlight the emergent two disjoint populations of neurons.

stored in working memory between STORE and RECALL commands. The expected delay between STORE and RECALL (see the top row of Table 1) scales the working memory time span that is required to solve this task. In the upper four rows of Table 1, a different fixed time constant for SFA neurons in the SNN is chosen. In the last two rows a power-law distribution of these time constants for SFA in the SNN, as well as a uniform distribution is considered. One sees that the resulting diversity of time constants for SFA yields about the same performance as a fixed choice of the time constant that is aligned with the required memory span of the task. However, a much larger time constant (see the row with $\tau_a = 8$ s in the column with an expected memory span of 200 ms or 2 s for the task) or a substantially smaller time constant (see the row with $\tau_a = 2$ s in the column with an expected memory span of 8 s) tends to work well.

Expected delay between STORE and RECALL	200 ms	2 s	4 s	8 s	16 s
without SFA ($\tau_a = 0$ ms)	96.7	51	50	49	51
$\tau_a = 200$ ms	99.92	73.6	58	51	51
$\tau_a = 2$ s	99.0	99.6	98.8	92.2	75.2
$\tau_a = 4$ s	99.1	99.7	99.7	97.8	90.5
$\tau_a = 8$ s	99.6	99.8	99.7	97.7	97.1
τ_a power-law dist. in [0, 8] s	99.6	99.7	98.4	96.3	83.6
τ_a uniform dist. in [0, 8] s	96.2	99.9	98.6	92.1	92.6

Table 1: Recall accuracy (in %) of SNN models with different time constants of SFA (rows) for variants of the STORE-RECALL task with different required memory time spans (columns). Good task performance does not require good alignment of SFA time constants with the required time span for working memory. An SNN consisting of 60 LIF neurons with SFA was trained for many different choices of SFA time constants for variations of the one-dimensional STORE-RECALL task with different required time spans for working memory. A network of 60 LIF neurons without SFA trained under the same parameters did not improve beyond chance level ($\sim 50\%$ accuracy), except for the task instance with an expected delay of 200 ms where the LIF network reached 96.7% accuracy (see top row).

SFA improves the performance of SNNs for common benchmark tasks that require computational operations on temporally dispersed information

An efficient model of temporal computing must not only be able to robustly store and recall information, but also actively manipulate the memory content. To investigate the impact of SFA in this context, we trained SNNs with and without SFA on a standard benchmark task for time series classification: the pixel-wise sequential MNIST (sMNIST) pattern classification task. In this variant of the well-known handwritten digit recognition data set MNIST, the pixels of each sample of a handwritten digit are temporally dispersed: they are given to the network one at a time, as they arise from a row-wise scanning pattern. This temporal computing task can apparently not be solved by just storing some bits in a working memory and recalling them later. Instead, evidence that speaks for or against the hypothesis that a particular digit is represented by the currently received time series must be updated continuously. This task also requires very good generalization capability, since the pixel sequences for different handwriting styles of the same digit may vary widely, and the network is tested on samples that were not used during optimization of the weights. For details see Methods and Supplement.

An SNN with SFA was able to solve this task with a test accuracy of 93.7%, whereas an SNN without SFA was only able to reach an accuracy of 51.8%. This demonstrates the significant impact of SFA on the ability of SNNs to actively manipulate information retained from previous inputs. See section 2 of the Supplement for more results with sparse connectivity, enforcement of Dale's Law and comparison to ANNs.

We also compared the performance of SNNs with and without SFA on the keyword spotting task Google Speech Commands Dataset (Warden, 2018) (v0.02). To solve this task the network

needs to learn to correctly disambiguate between audio recordings of silence, unknown words, or one of ten keywords. On this task, the performance of SNNs increases with the inclusion of SFA (from 89.04% to 91.21%) and approaches the state-of-the-art artificial recurrent model (93.18%), see section 3 of the Suppl. and Table S1.

Finally, we tested the performance of SNNs with SFA on the delayed-memory XOR task, because this task has previously already been used as benchmark tasks for SNNs in (Huh and Sejnowski, 2018). In this task, the network is required to compute the exclusive-or operation on the history of input pulses when prompted by a go-cue signal. Across 10 different runs, an SNN with SFA solved the task with $95.19 \pm 0.014\%$ accuracy, whereas the SNN without SFA converged at a much lower accuracy of $61.30 \pm 0.029\%$, see section 4 of the Suppl. and Fig. S3.

The good performance of SNNs with SFA on all three tasks demonstrates the powerful temporal computation capability of these networks.

SFA supports demanding cognitive computations on sequences with dynamically changing rules

The 12AX task — which can be viewed as a simplified version of the Wisconsin Card Sorting task (Berg, 1948) — tests the capability of subjects to apply dynamically changing rules for detecting specific subsequences in a long sequence of symbols as target sequences, and to ignore currently irrelevant inputs (O'Reilly and Frank, 2006; MacDonald III, 2008). It also probes the capability to maintain and update a hierarchical working memory, since the currently active rule — the context — stays valid for a longer period of time, and governs what other symbols should be stored in working memory.

More precisely, after processing any symbol in the sequence, the network should output "R" if this symbol terminates a context-dependent target sequence and "L" otherwise. The current target sequence depends on the current context, which is defined through the symbols "1" and "2". If the most recently received digit was a "1", the subject should output "R" only when it encounters a symbol "X" that terminates a subsequence A...X. This occurs, for example, for the 7th symbol in the trial shown in Fig. 3. In case that the most recent input digit was a "2", the subject should instead respond "R" only after the symbol "Y" in a subsequent subsequence B...Y (see the 20th symbol in Fig. 3). In addition, the processed sequence contains letters "C" and "Z" that are irrelevant and serve as distractors. This task requires a hierarchical working memory because the most recently occurring digit determines whether subsequent occurrences of "A" or "B" should be placed into working memory. Note also that neither the content of the higher-level working memory — the digit — nor the content of the lower-level working memory — the letter A or B — are simply recalled. Instead, they affect the target outputs of the network in a more indirect way. Furthermore, the higher-level processing rule affects what is to be remembered at the lower level.

A simpler version of this task, where X and Y were relevant only if they directly followed A or B respectively, and where fewer irrelevant letters occurred in the input, was solved in (O'Reilly and Frank, 2006; Martinolli et al., 2018; Kruijne et al., 2020) through biologically inspired artificial neural network models that were endowed with special working memory modules. Note that for this simpler version no lower-order working memory is needed, because one just has to wait for an immediate transition from A to X in the input sequence, or for an immediate transition from B to Y. But neither the simpler nor the more complex version, that is considered here, of the 12AX task has previously been solved by a network of spiking neurons.

In the version of the task that we consider, the distractor symbols between relevant symbols occur rather frequently. Hence robust maintenance of relevant symbols in the hierarchical working memory becomes crucial, because time spans between relevant symbols become longer, and hence the task is more demanding — especially for a neural network implementation.

Overall, the network received during each trial (episode) sequences of 90 symbols from the set {1, 2, A, B, C, X, Y, Z}, with repetitions as described in Methods. See the top of Fig. 3 for an example (the context-relevant symbols are marked in bold for visual ease).

We show in Fig. 3 that a generic SNN with SFA can solve this quite demanding version of the 12AX task. The network consisted of 200 recurrently connected spiking neurons (100 with and 100 without SFA), with all-to-all connections between them. After training, for new symbol sequences

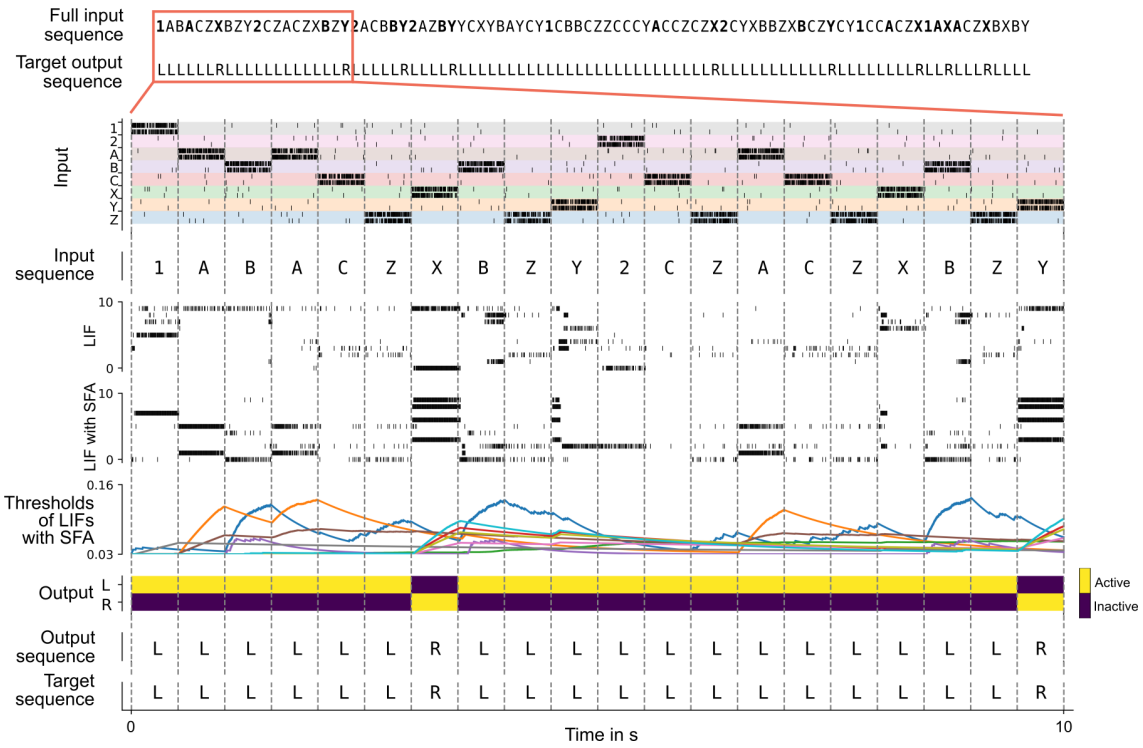

Figure 3: Solving the 12AX task by a network of spiking neurons with SFA. A sample trial of the trained network is shown. From top to bottom: Full input and target output sequence for a trial, consisting of 90 symbols each, blow-up for a subsequence of the input symbols, firing activity of 10 sample LIF neurons without and 10 sample neurons with SFA from the network, time course of the firing thresholds of these neurons with SFA, activation of the two readout neurons, the resulting sequence of output symbols which the network produced, and the target output sequence.

that had never occurred during training, the network produced an output string with all correct symbols in 97.79% of episodes. In contrast, a recurrent SNN with the same architecture but no neurons with SFA could achieve only 0.39% fully correct output strings (not shown).

Surprisingly, it was not necessary to create a special network architecture for the two levels of working memory that our more complex version of the 12AX task requires: A near perfectly performing network emerged from training a generic SNN with SFA. This shows that neurons with SFA enable generic recurrent networks of spiking neurons to solve demanding cognitive tasks involving dynamically changing rules and two levels of working memory.

SFA enables SNNs to carry out complex operations on sequences of symbols

Learning to carry out operations on sequences of symbols in such a way that they generalize to new sequences is a fundamental capability of the human brain, but a generic difficulty for neural networks (Marcus, 2003). Not only humans, but also non-human primates are able to carry out operations on sequences of items, and numerous neural recordings — starting with (Barone and Joseph, 1989) up to recent results such as (Carpenter et al., 2018; Liu et al., 2019) — provide information about the neural codes for sequences that accompany such operations in the brain. One fundamental question is how serial order of items is encoded in working memory. Behind this is the even more basic question of how transient structural information — the serial position of an item — is combined in the brain with content information about the identity of the item (Lashley, 1951). Obviously, this question also lies at the heart of open questions about the interplay between neural codes for syntax and semantics that enable language understanding in the human brain. The experimental data both of (Barone and Joseph, 1989) and (Liu et al., 2019) suggest that the brain uses a factorial code where position and identity of an item in a sequence are encoded separately by some neurons, thereby facilitating flexible generalization of learned experience to new sequences.

We show here that SNNs with SFA can be trained to carry out complex operations on sequences, are able to generalize such capabilities to new sequences, and produce spiking activity and neural codes that can be compared with neural recordings from the brain. In particular, they also produce factorial codes, where separate neurons encode the position and identity of a symbol in a sequence. One basic operation on sequences of symbols is remembering and reproducing a given sequence (Liu et al., 2019). This task had been proposed by (Marcus, 2003) to be a symbolic computation task that is fundamental for symbol processing capabilities of the human brain. But non-human primates can also learn simpler versions of this task, and hence one had been able to analyze how neurons in the brain encode the position and identity of symbols in a sequence (Barone and Joseph, 1989; Carpenter et al., 2018). A more complex operation that can also be carried out by the human brain is the reversal of a sequence (Marcus, 2003; Liu et al., 2019). We show that an SNN with SFA can carry out both of these operations, and is able to apply them to new sequences of symbols that did not occur during the training of the network.

We trained an SNN consisting of 320 recurrently connected LIF neurons (192 with and 128 without SFA) to carry out these two operations on sequences of 5 symbols from a repertoire of 31 symbols. After training, the SNN with SFA was able to apply duplication and reversal to new sequences also, achieving a success rate of 95.88% (average over 5 different network initializations) for unseen sequences. The "success rate" was defined as the fraction of test episodes (trials) where the full output sequence was generated correctly. Sample episodes of the trained SNN are shown in Fig. 4A. For comparison, we also trained a LIF network without SFA in exactly the same way with the same number of neurons. It achieved a performance of 0.0%.

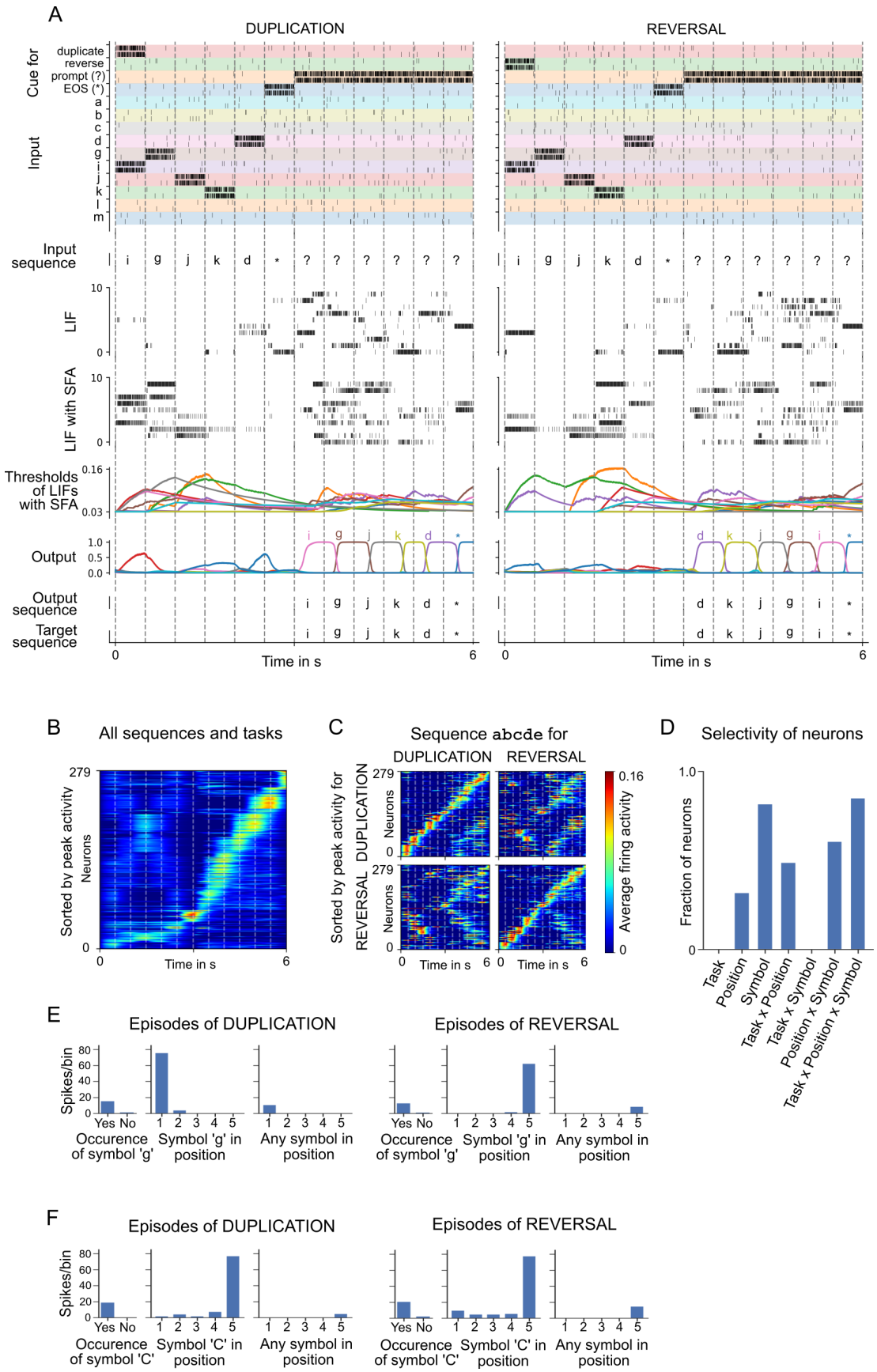

Figure 4: (Caption on the next page.)

Figure 4: Analysis of an SNN with SFA trained to carry out operations on sequences. (**A**) Two sample episodes where the network carried out sequence duplication (left) and reversal (right). Top to bottom: Spike inputs to the network (subset), sequence of symbols they encode, spike activity of 10 sample LIF neurons (without and with SFA) in the SNN, firing threshold dynamics for these 10 LIF neurons with SFA, activation of linear readout neurons, output sequence produced by applying argmax to them, target output sequence. (**B**-**F**) Emergent neural coding of 279 neurons in the SNN, and Peri-Condition Time Histogram (PCTH) plots of two sample neurons. Neurons are sorted by time of peak activity. (**B**) A substantial number of neurons were sensitive to the overall timing of the tasks, especially for the second half of trials when the output sequence is produced. (**C**) Neurons separately sorted for duplication episodes (left column) and reversal episodes (right column). Many neurons responded to input symbols according to their serial position, but differently for different tasks. (**D**) Histogram of neurons categorized according to conditions with statistically significant effect (3-way ANOVA). Firing activity of a sample neuron that fired primarily when: (**E**) the symbol "g" was to be written at the beginning of the output sequence. The activity of this neuron depended on the task context during the input period; (**F**) the symbol "C" occurred in position 5 in the input, irrespective of the task context.

A diversity of neural codes in SNNs with SFA. Emergent coding properties of neurons in the SNN are analyzed in Fig. 4B-F. Neurons are sorted in Fig. 4B,C according to the time of their peak activity (averaged over 1000 episodes), like in (Harvey et al., 2012). A number of network neurons (about one-third) participate in sequential firing activity independent of the type of task and the symbols involved (Fig. 4B). Instead, these neurons have learned to abstract the overall timing of the tasks. This kind of activity is reminiscent of the neural activity relative to the start of a trial that was recorded in rodents after they had learned to solve tasks that had a similar duration (Tsao et al., 2018).

The time of peak activity of other neurons depended on the task and the concrete content, see Fig. 4C. Interestingly enough, these neurons change their activation order already during the loading of the input sequence in dependence of the task (duplication or reversal). Using 3-way ANOVA, we were able to categorize each neuron as selective to a specific condition or a non-linear combination of conditions based on the effect size ω^2. Each neuron could belong to more than one category if the effect size was above the threshold of 0.14 (as suggested by (Field, 2013)). Similar to recordings from the brain (Carpenter et al., 2018), a diversity of neural codes emerged that encode one or several of the variables: symbol identity, serial position in the sequence, and type of task. In other words, a large fraction of neurons were mixed-selective, i.e. selective to non-linear combinations of all three variables. Peri-Condition Time Histogram (PCTH) plots of two sample neurons are shown in Fig. 4E,F: One neuron is selective to symbol "g" but at different positions depending on task context; The other neuron is selective to symbol "C" occurring at position 5 in the input, independent of task context. Thus one sees that a realization of this task by an SNN, which was previously not available, provides rich opportunities for a comparison of emergent spike codes in the model and neuronal recordings from the brain.

Discussion

Brains are able to carry out complex computations on temporally dispersed information, for example, on visual inputs streams, or on sequences of symbols. Previous neural network solutions for similarly demanding temporal computing tasks were based on artificial Long Short-Term Memory (LSTM) units. These LSTM units are commonly used in machine learning, but they cannot readily be mapped to units of neural networks in the brain. Hence it had remained an open problem how spiking neural networks of the brain could carry out such computations. We have shown that by adding to the standard model for SNNs one important feature of a substantial fraction of neurons in the neocortex, SFA, SNNs become able to solve such demanding temporal computing tasks.

In particular, we have shown that SNNs that contain neurons with SFA are able to carry out the 12AX task (Fig. 3). This task has been used to test the capability of human subjects to make online decisions while receiving sequences of symbols. These online decisions have to be carried

out according to rules that are also encoded — and occasionally changed — by symbols that occur in the same sequence of symbols.

We have shown in Fig. 4 that SNNs that contain neurons with SFA can solve another cognitively demanding task on temporally dispersed information: reproducing a previously received symbol sequence, or inverting its order. Since our model consists of spiking neurons, one can directly compare emergent neural codes for the identity and sequence positions of previously encountered symbols with recordings from neurons in the monkey brain for corresponding behavioral tasks. Similarly as in the recordings of (Barone and Joseph, 1989; Carpenter et al., 2018), a diversity of neural codes emerge in the model, where neurons encode one or several of the relevant variables – symbol identity, serial position of a symbol, and type of task.

We have also elucidated the computational principle that enables neurons with SFA to support temporal computing tasks. We had considered for that purpose in Fig. 2 a simpler temporal computing task where no intermediate updating of stored information was required: All preceding information was provided at a single time point A, and had to be recalled at a well-specified later time point B. We found there that neurons with SFA store information through negative imprinting, i.e., neurons that fire more during time point A fire less at time point B. Hence they encode past information in an activity-silent manner. Such activity-silent form of working memory has been found in the human brain for content that is currently not in the focus of attention (Wolff et al., 2017). A prediction of the negative imprinting principle is that decoders that are trained to decode information at time point A are not able to decode this information during a network reactivation at some intermediate time point C between time points A and B, and vice versa (see subsection "Decoding memory from the network activity" in Methods). This prediction had been verified for the human brain in the experiments of (Wolff et al., 2017). It is actually well-known that negative imprinting is used by the brain for a particular type of long-term memory called recognition memory: Familiarity of an object is encoded through the reduced firing of a large fraction of neurons in the perirhinal cortex and adjacent areas, see (Winters et al., 2008) for a review.

Our model makes a number of concrete suggestions for further experiments. It suggests that a refined decoder that takes negative imprinting into account is able to elucidate the transformation of stored information between time points A (encoding) and an intermediate time point C (network reactivation) in the experiment of (Wolff et al., 2017).

Furthermore, the strong role of SFA for temporal computing tasks that we found predicts that SFA is more common in those brain areas that play a key role in temporal computing tasks. At the same time, our detailed analysis of the required alignment between the time scale of SFA and the time scale of working memory duration can be rather loose. Even a random distribution of time constants for SFA works well. Previous experiments have already reported history-dependence of neural firing for up to 20 s (Pozzorini et al., 2013, 2015), but a more systematic analysis is needed.

Finally, our results raise the question of the extent to which the distribution of time scales of neurons with SFA in a cortical area is related to the intrinsic time scale of that cortical area as measured via intrinsic fluctuations of spiking activity (Murray et al., 2014; Wasmuht et al., 2018). Are neurons with SFA essential for defining the intrinsic time scale of an area, or can one find cases where both time scales diverge? We tested the relation between time constants of SFA and the intrinsic time scale of neurons for the case of the STORE-RECALL task (see section 1 of the Suppl. and Fig. S1). We found that the time constants of neurons with SFA had little impact on their intrinsic time scale for this task. We conjecture that the network input has a stronger impact on the intrinsic time scales of neurons for this task.

Altogether, we have shown that a well-known feature of a substantial fraction of neurons in the neocortex — SFA — provides an important new facet for our understanding of computations in SNNs: It enables SNNs to integrate temporally dispersed information seamlessly into ongoing network computations. This paves the way for reaching a key-goal of modeling — to combine detailed experimental data from neurophysiology on the level of neurons and synapses with the brain-like high computational performance of the network.

Materials and Methods

Table of Contents:

- Network models
- Training method
- Tasks

Network models

Leaky integrate and fire (LIF) neurons. A LIF neuron j spikes as soon at its membrane potential $V_j(t)$ is above its threshold v_{th}. At each spike time t, the membrane potential $V_j(t)$ is reset by subtracting the threshold value v_{th} and the neuron enters a strict refractory period for 3 to 5 ms (depending on the experiment) where it cannot spike again. Between spikes, the membrane voltage $V_j(t)$ is following the dynamic:

$$\tau_m \dot{V}_j(t) \quad = \quad -V_j(t) + R_m I_j(t),$$

where τ_m is the membrane constant of neuron j, R_m is the resistance of the cell membrane, and I_j the input current.

Our simulations were performed in discrete time with a time step $\delta t = 1\,\text{ms}$. In discrete time, the input and output spike trains are modeled as binary sequences $x_i(t), z_j(t) \in \{0, \frac{1}{\delta t}\}$ respectively. Neuron j emits a spike at time t if it is currently not in a refractory period, and its membrane potential $V_j(t)$ is above its threshold. During the refractory period following a spike, $z_j(t)$ is fixed to 0. The neural dynamics in discrete time reads as follows:

$$V_j(t+\delta t) \quad = \quad \alpha V_j(t) + (1-\alpha) R_m I_j(t) - v_{\text{th}} z_j(t) \delta t, \tag{2}$$

where $\alpha = \exp(-\frac{\delta t}{\tau_m})$, with τ_m being the membrane constant of the neuron j. The spike of neuron j is defined by $z_j(t) = H\left(\frac{V_j(t)-v_{\text{th}}}{v_{\text{th}}}\right)\frac{1}{\delta t}$, with $H(x) = 0$ if $x < 0$ and 1 otherwise. The term $-v_{\text{th}} z_j(t)\delta t$ implements the reset of the membrane voltage after each spike.

In all simulations, the R_m was set to $1\,\text{G}\Omega$. The input current $I_j(t)$ is defined as the weighted sum of spikes from external inputs and other neurons in the network:

$$I_j(t) \quad = \quad \sum_i W_{ji}^{in} x_i(t - d_{ji}^{in}) + \sum_i W_{ji}^{rec} z_i(t - d_{ji}^{rec}), \tag{3}$$

where W_{ji}^{in} and W_{ji}^{rec} denote respectively the input and the recurrent synaptic weights and d_{ji}^{in} and d_{ji}^{rec} the corresponding synaptic delays.

LIF neurons with SFA. The SFA is realized by replacing the fixed threshold v_{th} with the adaptive threshold $A_j(t)$ which follows the dynamic (reproducing equation (1) for arbitrary δt):

$$\begin{aligned} A_j(t) &= v_{\text{th}} + \beta a_j(t), \\ a_j(t+\delta t) &= \rho_j a_j(t) + (1-\rho_j) z_j(t) \delta t, \end{aligned} \tag{4}$$

Now, the parameter ρ_j is given by $\rho_j = \exp\left(\frac{-\delta t}{\tau_{a,j}}\right)$. In all our simulations, δt was set to 1 ms.

The spiking output of LIF neuron with SFA j is then defined by $z_j(t) = H\left(\frac{V_j(t)-A_j(t)}{A_j(t)}\right)\frac{1}{\delta t}$.

Adaptation time constants of neurons with SFA were chosen to match the task requirements while still conforming to the experimental data from rodents (Allen Institute, 2018; Pozzorini et al., 2013, 2015; Mensi et al., 2012). For an analysis of the impact of the adaptation time constants on the performance see Table 1.

Weight initialization. Initial input and recurrent weights were drawn from a Gaussian distribution $W_{ji} \sim \frac{w_0}{\sqrt{n_{in}}}\mathcal{N}(0,1)$, where n_{in} is the number of afferent neurons and $\mathcal{N}(0,1)$ is the zero-mean unit-variance Gaussian distribution and $w_0 = \frac{1\,\text{Volt}}{R_m}\delta t$ is a normalization constant (Bellec et al., 2018).

Training method

In artificial recurrent neural networks, gradients can be computed with backpropagation through time ($BPTT$). In spiking neural networks, complications arise from the non-differentiability of the output of spiking neurons. In our discrete-time simulation, this is formalized by the discontinuous step function H arising in the definition of the spike variable $z_j(t)$. All other operations can be differentiated exactly with $BPTT$. For feedforward artificial neural networks using step functions, a solution was to use a pseudo derivative $H'(x) := \max\{0, 1 - |x|\}$, but we observed that this convention is unstable with recurrently connected neurons. We found that dampening this pseudo-derivative with a factor $\gamma < 1$ (typically $\gamma = 0.3$) solves that issue. Hence we use the pseudo-derivative:

$$\frac{dz_j(t)}{dv_j(t)} := \gamma \max\{0, 1 - |v_j(t)|\}, \tag{5}$$

where $v_j(t)$ denotes the normalized membrane potential $v_j(t) = \frac{V_j(t) - A_j(t)}{A_j(t)}$. Importantly, gradients can propagate in adaptive neurons through many time steps in the dynamic threshold without being affected by the dampening.

Tasks

20-dimensional STORE-RECALL task. The input to the network consisted of commands STORE and RECALL, and 20 bits which were represented by subpopulations of spiking input neurons. STORE and RECALL commands were represented by 4 neurons each. The 20 bits were represented by population coding where each bit was assigned 4 input neurons (2 for value zero, and 2 for value one). When a subpopulation was active, it would exhibit a Poisson firing with a frequency of 400 Hz. Each input sequence consisted of 10 steps (200 ms each) where a different population encoded bit string was shown during every step. Only during the RECALL period, the input populations, representing the 20 bits, were silent. At every step, the STORE or the RECALL populations were activated interchangeably with probability 0.2 which resulted in the distribution of delays between the STORE-RECALL pairs in the range [200, 1600] ms.

To measure the generalization capability of a trained network, we first generated a test set dictionary of 20 unique feature vectors (random bit strings of length 20) that had at least a Hamming distance of 5 bits among each other. For every training batch, a new dictionary of 40 random bit strings (of length 20) was generated where each string had a Hamming distance of at least 5 bits from any of the bit string in the test set dictionary. This way we ensured that, during training, the network never encountered any bit string similar to one from the test set.

Networks were trained for 4000 iterations with a batch size of 256 and stopped if the error on the training batch was below 1%. We used Adam optimizer (Kingma and Ba, 2014) with default parameters and initial learning rate of 0.01 which is decayed every 200 iterations by a factor of 0.8. We also used learning rate ramping, which, for the first 200 iterations, monotonically increased the learning rate from 0.00001 to 0.01. To avoid unrealistically high firing rates, the loss function contained a regularization term (scaled with coefficient 0.001) that minimizes the squared difference of the average firing rate of individual neurons from a target firing rate of 10 Hz. To improve convergence, we also included an entropy component to the loss (scaled with coefficient 0.3) which was computed as the mean of the entropies of the outputs of the sigmoid neurons. The test performance was computed as average over 512 random input sequences.

We trained SNNs with and without SFA, consisting of 500 recurrently connected neurons. The membrane time constant was $\tau_m = 20$ ms and the refractory period was 3 ms. Adaptation parameters were $\beta = 4$ mV and $\tau_a = 800$ ms with baseline threshold voltage 10 mV. The synaptic delay was 1 ms. The input to the sigmoidal readout neurons were the neuron traces that were calculated by passing all the network spikes through a low-pass filter with a time constant of 20 ms.

We ran 5 training runs with different random seeds (initializations) for both SNNs with and without SFA. All runs of the SNN with SFA network converged after ~ 3600 iterations to a training error below 1%. At that point we measured the accuracy on 512 test sequences generated using the previously unseen test bit strings which resulted in test accuracy of 99.09% with a standard

deviation of 0.17%. The LIF network was not able to solve the task in any of the runs (all runs resulted in 0% training and test accuracy with zero standard deviation). On the level of individual feature recall accuracy, the best one out of 5 training runs of the LIF network was able to achieve 49% accuracy which is the chance level since individual features are binary bits. In contrast, all SNNs with SFA runs had individual feature level accuracy of above 99.99%.

One-dimensional STORE-RECALL task. The input to the network consisted of 40 input neurons: 10 for STORE, 10 for RECALL, and 20 for population coding of a binary feature. Whenever a subpopulation was active, it would exhibit a Poisson firing with a frequency of 50 Hz. The input sequences of experiments with the expected delay of 2, 4, 8, and 16 s were constructed as a sequence of 20, 40, 80, 120 steps respectively, with each step lasting for 200 ms. For the experiment with expected delay of 200 ms, the input sequence consisted of 12 steps of 50 ms.

Networks were trained for 400 iterations with a batch size of 64.We used Adam optimizer with default parameters and initial learning rate of 0.01 which was decayed every 100 iterations by a factor of 0.3. The same firing rate regularization term was added to the loss as in the 20-dimensional STORE-RECALL setup (see above). The test performance was computed as the batch average over 2048 random input sequences.

Networks consisted of 60 recurrently connected neurons. The membrane time constant was $\tau_m = 20\,\text{ms}$ and the refractory period was 3 ms. Adaptation parameters were $\beta = 1\,\text{mV}$ with baseline threshold voltage 10 mV. Table 1 defines the adaptation time constants and expected delay of the experiments in that section. The synaptic delay was 1 ms. The same sigmoidal readout neuron setup was used as in the 20-dimensional STORE-RECALL setup (see above).

Decoding memory from the network activity. We trained a Support Vector Machine (SVM) to classify the stored memory content from the network spiking activity in the step before the RECALL (200 ms before the start of RECALL command). We performed a cross-validated grid-search to find the best hyperparameters for the SVM which included kernel type {linear, polynomial, RBF} and penalty parameter C of the error term {0.1, 1, 10, 100, 1000}. We trained SVMs on test batches of the 5 different training runs of 20-dimensional STORE-RECALL task. SVMs trained on the period preceding the RECALL command of a test batch achieved an average of 4.38% accuracy with a standard deviation of 1.29%. In contrast, SVMs trained on a period during the RECALL command achieved an accuracy of 100%. This demonstrates that the memory stored in the network is not decodable from the network firing activity before the RECALL input command.

Additionally, analogous to the experiments of (Wolff et al., 2017), we trained SVMs on network activity during the encoding (STORE) period and evaluated them on the network activity during reactivation (RECALL), and vice versa. In both scenarios, the classifiers were not able to classify the memory content of the evaluation period (0.0% accuracy).

sMNIST task. The input consisted of sequences of 784 pixel values created by unrolling the handwritten digits of the MNIST dataset, one pixel after the other in a scanline manner as indicated in Fig. S2A. We used 1 ms presentation time for each pixel gray value. Each of the 80 input neurons was associated with a particular threshold for the grey value, and this input neuron fired whenever the grey value crossed its threshold in the transition from the previous to the current pixel.

Networks were trained for 36,000 iterations using the Adam optimizer with batch size 256. The initial learning rate was 0.01 and every 2500 iterations the learning rate was decayed by a factor of 0.8. The same firing rate regularization term was added to the loss as in the STORE-RECALL setup (see above) but with the scaling coefficient of 0.1.

Networks consisted of 220 neurons. The network with SFA had 100 neurons out of 220 with SFA and the rest without. The neurons had a membrane time constant of $\tau_m = 20\,\text{ms}$, a baseline threshold of $v_{\text{th}} = 10\,\text{mV}$, and a refractory period of 5 ms. LIF neurons with SFA had the adaptation time constant $\tau_a = 700\,\text{ms}$ with adaptation strength $\beta = 1.8\,\text{mV}$. The synaptic delay was 1 ms. The output of the SNN was produced by the softmax of 10 linear output neurons that received the low-pass filtered version of the spikes from all neurons in the network, as shown in the bottom row of Fig. S2B. The low-pass filter had a time constant of 20 ms. For training the network to classify into one of the ten classes we used cross-entropy loss computed between the

labels and the softmax of output neurons.

The 12AX task. The input for each training and testing episode consisted of a sequence of 90 symbols from the set {1,2,A,B,C,X,Y,Z}. A single episode could contain multiple occurrences of digits 1 or 2 (up to 23), each time changing the target sequence (A...X or B...Y) after which the network was supposed to output R. Each digit could be followed by up to 26 letters before the next digit appeared. More precisely, the following regular expression describes the string that was produced: `[12][ABCXYZ]{1,10}((A[CZ]{0,6}X|B[CZ]{0,6}Y)|([ABC][XYZ])){1,2}`. Each choice in this regular expression was made randomly.

The network received spike trains from the input population of spiking neurons, producing Poisson spike trains. Possible input symbols were encoded using one-hot coding. Each input symbol was signaled through a high firing rate of a separate subset of 5 input neurons for 500 ms. The output consisted of two readouts, one for L, one for the R response. During each 500 ms time window, the input to these readouts was the average activity of neurons in the SNN during that time window. The final output symbol was based on which of the two readouts had the maximum value.

The neurons had a membrane time constant of $\tau_m = 20\,\text{ms}$, a baseline threshold $v_{\text{th}} = 30\,\text{mV}$, a refractory period of 5 ms, and synaptic delays of 1 ms. LIF neurons with SFA had an adaptation strength of $\beta = 1.7\,\text{mV}$, and adaptation time constants were chosen uniformly from $[1, 13500]$ ms.

A cross-entropy loss function was used along with a regularization term (scaled with coefficient 15) that minimizes the squared difference of average firing rate between individual neurons and a target firing rate of 10 Hz. The SNN was trained using the Adam optimizer for 10,000 iterations with a batch size of 20 episodes and a fixed learning rate of 0.001. An episode consisted of 90 steps, with between 4 to 23 tasks generated according to the task generation procedure described previously. We trained the network with *BPTT* using 5 different network initializations, which resulted in an average test success rate of 97.79% with a standard deviation of 0.42%.

Symbolic computation on strings of symbols. The input to the network consisted of 35 symbols: 31 symbols represented symbols from the English alphabet {a, b, c, d, ... x, y, z, A, B, C, D, E}, one symbol was for "end-of-string" (EOS) '*', one for cue for the output prompt '?', and two symbols to denote whether the task command was duplication or reversal. Each of the altogether 35 input symbols were given to the network in the form of higher firing activity of a dedicated population of 5 input neurons outside of the SNN ("one-hot encoding"). This population of input neurons fired at a "high" rate (200 Hz) to encode 1, and at a "low" rate (2 Hz) otherwise. The network output was produced by linear readouts (one per potential output symbol, each with a low pass filter with a time constant of 250 ms) that received spikes from neurons in the SNN, see the row "Output" in Fig. 4A. The final output symbol was selected using the readout which had the maximum value at the end of each 500 ms time window (a softmax instead of the hard argmax was used during training), mimicking winner-take-all computations in neural circuits of the brain (Chettih and Harvey, 2019) in a qualitative manner.

The network was trained to minimize the cross-entropy error between the softmax applied to the output layer and targets. The loss function contained a regularization term (scaled with coefficient 5) that minimizes the squared difference of average firing rate between individual neurons and a target firing rate of 20 Hz.

The training was performed for 50,000 iterations, with a batch size of 50 episodes. We used Adam optimizer with default parameters and a fixed learning rate of 0.001. Each symbol was presented to the network for a duration of 500 ms. The primary metric we used for measuring the performance of the network was success rate, which was defined as the percentage of episodes where the network produced the full correct output for a given string i.e. all the output symbols in the episode had to be correct. The network was tested on 50,000 previously unseen strings.

The network consisted of 192 LIF neurons with SFA and 128 LIF neurons without SFA. All the neurons had a membrane time constant of $\tau_m = 20\,\text{ms}$, a baseline threshold $v_{\text{th}} = 30\,\text{mV}$, a refractory period of 5 ms, and a synaptic delay of 1 ms. LIF neurons with SFA in the network had an adaptation strength of $\beta = 1.7\,\text{mV}$. It was not necessary to assign particular values to adaptation time constants of firing thresholds of neurons with SFA; we simply chose them uniformly randomly to be between 1 ms and 6000 ms, mimicking the diversity of SFA effects found in the

neocortex (Allen Institute, 2018) in a qualitative manner. All other parameters were the same as in the other experiments. We trained the network using 5 different network initializations (seeds) and tested it on previously unseen strings. Average test success rate was 95.88% with standard deviation 1.39%.

Analysis of spiking data. We used 3-way ANOVA to analyze if a neuron's firing rate is significantly affected by task, serial position in the sequence, symbol identity, or combination of these (similar to (Lindsay et al., 2017)). We refer to these factors as "conditions". The analysis was performed on the activity of the neurons of the trained SNN during 50,000 test episodes. For the analysis, neurons whose average firing rate over all episodes was lower than 2Hz or greater than 60Hz were discarded from the analysis to remove large outliers. This left 279 out of the 320 neurons. From each episode, a serial position from the input period was chosen randomly, and hence each episode could be used only once, i.e., as one data point. This was to make sure that each entry in the 3-way ANOVA was completely independent of other entries, since the neuron activity within an episode is highly correlated. Each data point was labelled with the corresponding triple of (task type, serial position, symbol identity). To ensure that the dataset was balanced, the same number of data points per particular combination of conditions was used, discarding all the excess data points, resulting in a total of 41,850 data points. To categorize a neuron as selective to one or more conditions, or combination of conditions, we observed p-values obtained from $3-$way ANOVA and calculated the effect size ω^2 for each combination of conditions. If the p-value was smaller than 0.001 and ω^2 greater than 0.14 for a particular combination of conditions, the neuron was categorized as selective to that combination of conditions. The ω^2 threshold of 0.14 was suggested by (Field, 2013) to select large effect sizes. Each neuron can have a large effect size for more than one combination of conditions. Thus the values shown in Fig. 4D sum to > 1. The neuron shown in Fig. 4E had the most prominent selectivity for the combination of Task $\times$ Position $\times$ Symbol, with $\omega^2 = 0.394$ and $p < 0.001$. The neuron shown in Fig. 4F was categorized as selective to a combination of Position $\times$ Symbol category, with $\omega^2 = 0.467$ and $p < 0.001$. While the 3-way ANOVA tells us if a neuron is selective to a particular combination of conditions, it does not give us the exact task/symbol/position that the neuron is selective to. To find the specific task/symbol/position that the neuron was selective to, Welch's t-test was performed, and a particular combination with maximum t-statistic and $p < 0.001$ was chosen to be shown in Fig. 4E,F.

References

Allen Institute. Allen Cell Types Database Technical white paper: GLIF models `http://help.brain-map.org/download/attachments/8323525/glifmodels.pdf`. Technical report, October 2017. v4.

Allen Institute. © 2018 Allen Institute for Brain Science. Allen Cell Types Database, cell feature search. Available from: `celltypes.brain-map.org/data`. 2018.

P. Barone and J.-P. Joseph. Prefrontal cortex and spatial sequencing in macaque monkey. *Experimental brain research*, 78(3):447–464, 1989.

G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Advances in Neural Information Processing Systems*, pages 787–797, 2018.

J. Benda and A. V. Herz. A universal model for spike-frequency adaptation. *Neural computation*, 15(11):2523–2564, 2003.

J. Benda, L. Maler, and A. Longtin. Linear versus nonlinear signal transmission in neuron models with adaptation currents or dynamic thresholds. *Journal of Neurophysiology*, 104(5):2806–2820, 2010.

E. A. Berg. A simple objective technique for measuring flexibility in thinking. *The Journal of general psychology*, 39(1):15–22, 1948.

A. F. Carpenter, G. Baud-Bovy, A. P. Georgopoulos, and G. Pellizzer. Encoding of serial order in working memory: neuronal activity in motor, premotor, and prefrontal cortex during a memory scanning task. *Journal of Neuroscience*, 38(21):4912–4933, 2018.

S. N. Chettih and C. D. Harvey. Single-neuron perturbations reveal feature-specific competition in V1. *Nature*, 567(7748):334–340, 2019.

S. Deneve. Bayesian spiking neurons i: inference. *Neural computation*, 20(1):91–117, 2008.

B. Ermentrout. Linearization of fi curves by adaptation. *Neural computation*, 10(7):1721–1729, 1998.

A. Field. *Discovering statistics using IBM SPSS statistics*. Sage, 2013.

H. Fitz, M. Uhlmann, D. Van den Broek, R. Duarte, P. Hagoort, and K. M. Petersson. Neuronal spike-rate adaptation supports working memory in language processing. *Proceedings of the National Academy of Sciences*, 117(34):20881–20889, 2020.

B. Gutkin and F. Zeldenrust. Spike frequency adaptation. *Scholarpedia*, 9(2):30643, 2014. doi: 10.4249/scholarpedia.30643. revision #143322.

C. D. Harvey, P. Coen, and D. W. Tank. Choice-specific sequences in parietal cortex during a virtual-navigation decision task. *Nature*, 484(7392):62–68, 2012.

D. Huh and T. J. Sejnowski. Gradient descent for spiking neural networks. In *Advances in Neural Information Processing Systems*, pages 1433–1443, 2018.

Z. P. Kilpatrick and B. Ermentrout. Sparse gamma rhythms arising through clustering in adapting neuronal networks. *PLoS Comput Biol*, 7(11):e1002281, 2011.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

W. Kruijne, S. M. Bohte, P. R. Roelfsema, and C. N. L. Olivers. Flexible working memory through selective gating and attentional tagging. *Neural Computation*, 0(0):1–40, 2020. doi: 10.1162/neco_a_01339. URL `https://doi.org/10.1162/neco_a_01339`. PMID: 33080159.

K. S. Lashley. *The problem of serial order in behavior*, volume 21. Bobbs-Merrill Oxford, United Kingdom, 1951.

G. W. Lindsay, M. Rigotti, M. R. Warden, E. K. Miller, and S. Fusi. Hebbian learning in a random network captures selectivity properties of the prefrontal cortex. *Journal of Neuroscience*, 37(45): 11021–11036, 2017.

Y. Liu, R. J. Dolan, Z. Kurth-Nelson, and T. E. Behrens. Human replay spontaneously reorganizes experience. *Cell*, 178(3):640–652, 2019.

A. W. MacDonald III. Building a clinically relevant cognitive task: case study of the ax paradigm. *Schizophrenia bulletin*, 34(4):619–628, 2008.

G. F. Marcus. *The Algebraic Mind: Integrating Connectionism and Cognitive Science*. MIT Press, 2003.

E. Marder, L. Abbott, G. G. Turrigiano, Z. Liu, and J. Golowasch. Memory from the dynamics of intrinsic membrane currents. *Proceedings of the national academy of sciences*, 93(24):13481–13486, 1996.

M. Martinolli, W. Gerstner, and A. Gilra. Multi-timescale memory dynamics extend task repertoire in a reinforcement learning network with attention-gated memory. *Frontiers in computational neuroscience*, 12:50–50, Jul 2018. ISSN 1662-5188. doi: 10.3389/fncom.2018.00050. URL `https://pubmed.ncbi.nlm.nih.gov/30061819`. 30061819[pmid].

S. Mensi, R. Naud, C. Pozzorini, M. Avermann, C. C. Petersen, and W. Gerstner. Parameter extraction and classification of three cortical neuron types reveals two distinct adaptation mechanisms. *Journal of neurophysiology*, 107(6):1756–1775, 2012.

J. D. Murray, A. Bernacchia, D. J. Freedman, R. Romo, J. D. Wallis, X. Cai, C. Padoa-Schioppa, T. Pasternak, H. Seo, D. Lee, et al. A hierarchy of intrinsic timescales across primate cortex. *Nature neuroscience*, 17(12):1661, 2014.

R. C. O'Reilly and M. J. Frank. Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia. *Neural computation*, 18(2):283–328, 2006.

C. Pozzorini, R. Naud, S. Mensi, and W. Gerstner. Temporal whitening by power-law adaptation in neocortical neurons. *Nature neuroscience*, 16(7):942, 2013.

C. Pozzorini, S. Mensi, O. Hagens, R. Naud, C. Koch, and W. Gerstner. Automated high-throughput characterization of single neurons by means of simplified spiking models. *PLoS computational biology*, 11(6), 2015.

C. Teeter, R. Iyer, V. Menon, N. Gouwens, D. Feng, J. Berg, A. Szafer, N. Cain, H. Zeng, M. Hawrylycz, et al. Generalized leaky integrate-and-fire models classify multiple neuron types. *Nature communications*, 9(1):1–15, 2018.

A. Tsao, J. Sugar, L. Lu, C. Wang, J. J. Knierim, M. B. Moser, and E. I. Moser. Integrating time from experience in the lateral entorhinal cortex. *Nature*, 561(7721):57–52, 2018.

G. G. Turrigiano, E. Marder, and L. Abbott. Cellular short-term memory from a slow potassium conductance. *Journal of neurophysiology*, 75(2):963–966, 1996.

X.-J. Wang. Calcium coding and adaptive temporal computation in cortical pyramidal neurons. *Journal of Neurophysiology*, 1998.

P. Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

D. F. Wasmuht, E. Spaak, T. J. Buschman, E. K. Miller, and M. G. Stokes. Intrinsic neuronal dynamics predict distinct functional roles during working memory. *Nature communications*, 9 (1):3499, 2018.

B. D. Winters, L. M. Saksida, and T. J. Bussey. Object recognition memory: neurobiological mechanisms of encoding, consolidation and retrieval. *Neuroscience & Biobehavioral Reviews*, 32 (5):1055–1070, 2008.

M. J. Wolff, J. Jochim, E. G. Akyürek, and M. G. Stokes. Dynamic hidden states underlying working-memory-guided behavior. *Nature Neuroscience*, 20(6):864, 2017.

Acknowledgements

General: We would like to thank Pieter Roelfsema and Chris Summerfield for detailed comments on an earlier version of the manuscript.

Funding: This research was partially supported by the Human Brain Project (Grant Agreement number 785907) and the SYNCH project (Grant Agreement number 824162) of the European Union. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Quadro P6000 GPU used for this research. Computations were carried out on the Human Brain Project PCP Pilot Systems at the Juelich Supercomputing Centre, which received co-funding from the European Union (Grant Agreement number 604102) and on the Vienna Scientific Cluster (VSC).

Author contributions: The study was conceived by DS, AS, GB, WM. The experiments were designed by DS, AS, GB, RL, WM and were conducted by DS, AS, CK. The manuscript was written by DS, AS, CK, RL, GB, WM.

Competing interests: The authors declare no competing interests.

Data and materials availability: An implementation of the network model in Tensorflow/Python is available at `https://github.com/IGITUGraz/LSNN-official/`. The sMNIST dataset is available at `https://www.tensorflow.org/datasets/catalog/mnist`. The google speech commands dataset is available at `https://storage.cloud.google.com/download.tensorflow.org/data/speech_commands_v0.02.tar.gz`.