MEMSORN: Self-organization of an inhomogeneous memristive hardware for sequence learning

Melika Payvand (melika@ini.uzh.ch)
University of Zurich and ETH Zurich https://orcid.org/0000-0001-5400-067X

Filippo Moro
2CEA-Leti, Universite Grenoble Alpes

Kumiko Nomura
Toshiba Corporation

Thomas Dalgaty
CEA-Leti, Universite Grenoble Alpes

Elisa Vianello
CEA-LETI, Universite Grenoble Alpes

Yoshifumi Nishi
Toshiba Corporation

Giacomo Indiveri
University of Zurich and ETH Zurich https://orcid.org/0000-0002-7109-1689

Article

Keywords: MEMSORN, resistive memory, device-circuit-algorithm, neural network

Posted Date: October 14th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-955484/v1

Version of Record: A version of this preprint was published at Nature Communications on October 2nd, 2022. See the published version at https://doi.org/10.1038/s41467-022-33476-6.

MEMSORN: Self-organization of an inhomogeneous memristive hardware for sequence learning

Melika Payvand[1,*,+], Filippo Moro[1,2,+], Kumiko Nomura[3], Thomas Dalgaty[2], Elisa Vianello[2], Yoshifumi Nishi[3], and Giacomo Indiveri[1]

[1]Institute for Neuroinformatics, University of Zurich and ETH Zurich, Zurich, Switzerland
[2]CEA-Leti, Université Grenoble Alpes, Grenoble, France
[3]Corporate Research & Development Center, Toshiba Corporation, Kawasaki, Japan
[*]melika@ini.uzh.ch
[+]these authors contributed equally to this work

ABSTRACT

Learning is a fundamental component for creating intelligent machines. Biological intelligence orchestrates synaptic and neuronal learning at multiple time-scales to *self-organize* populations of neurons for solving complex tasks. Inspired by this, we design and experimentally demonstrate an adaptive hardware architecture *Memristive Self-organizing Spiking Recurrent Neural Network (MEMSORN)*. MEMSORN incorporates resistive memory (RRAM) in its synapses and neurons which configure their state based on Hebbian and Homeostatic plasticity respectively. For the first time, we derive these plasticity rules directly from the statistical measurements of our fabricated RRAM-based neurons and synapses. These "technologically plausible" learning rules exploit the intrinsic variability of the devices and improve the accuracy of the network on a sequence learning task by 30%. Finally, we compare the performance of MEMSORN to a fully-randomly set-up recurrent network on the same task, showing that self-organization improves the accuracy by more than 15%. This work demonstrates the importance of the device-circuit-algorithm co-design approach for implementing brain-inspired computing hardware.

Introduction

The hallmark of intelligence is the ability of the brain to adapt and *self-organize* itself to sensory information it receives throughout its lifespan. This self-organization is mediated by a rich set of neuro-cognitive mechanisms that together contribute to sequence learning and long-term memory formation[1]. While learning, a web of memory forms between large groups of neurons, leading to coherent dynamic activity patterns that are a function of the sensory information the system receives.

It has been shown that the combination of brain-inspired learning rules at different time-scales lend themselves to the self-organization of dynamic networks for behavior control[2,3]. This type of self-organization lies in the unsupervised learning realm where the ground truth is not available for learning. Instead, the memory forms as a result of clustering information in *cell-assemblies*[2]. A cell-assembly can be defined as a group of neurons with strong mutual excitatory connections. Once a subset of a cell-assembly is stimulated, its neurons tend to be activated as a whole, so that the cell can be considered as an operational unit of a Spiking Recurrent Neural Network (SRNN). Applying local learning rules to the recurrent connections forms independent cell-assemblies and makes the SRNN more powerful in extracting temporal features in the data, compared to a fully-randomly-connected solution[3]. One example of such a concept has been shown in Self-Organizing Recurrent Network (SORN)[3], a recurrent network model of excitatory and inhibitory binary neurons. It incorporates a Hebbian-based synaptic plasticity at a shorter timescale, along with Homeostatic plasticity at a longer timescale. It is illustrated that SORN outperforms random Recurrent Neural Networks (RNNs) without plasticity on sequence prediction tasks.

Implementing SORN-like networks on a hardware substrate holds great promise for machine intelligence and autonomous agents, especially in situations where the agent is in unknown environments[4,5]. Neuromorphic technologies with online learning capabilities can support the hardware implementation of such self-organizing SRNNs[6,7]. For example, a recent study has successfully mapped a self-organizing network on Loihi digital neuromorphic hardware for the generation of robust trajectories[8].

On-line learning in electronic devices requires local and distributed memory elements for storing the learned parameters (e.g., the synaptic weights). Resistive Random Access Memory (RRAM) has recently gained significant attention as a promising memory technology for on-line learning[9–17]. Its non-volatile and multi-state properties makes it a plausible candidate for employment as adaptive hardware. Importantly, its internal dynamics and intrinsic stochasticity have been proven beneficial

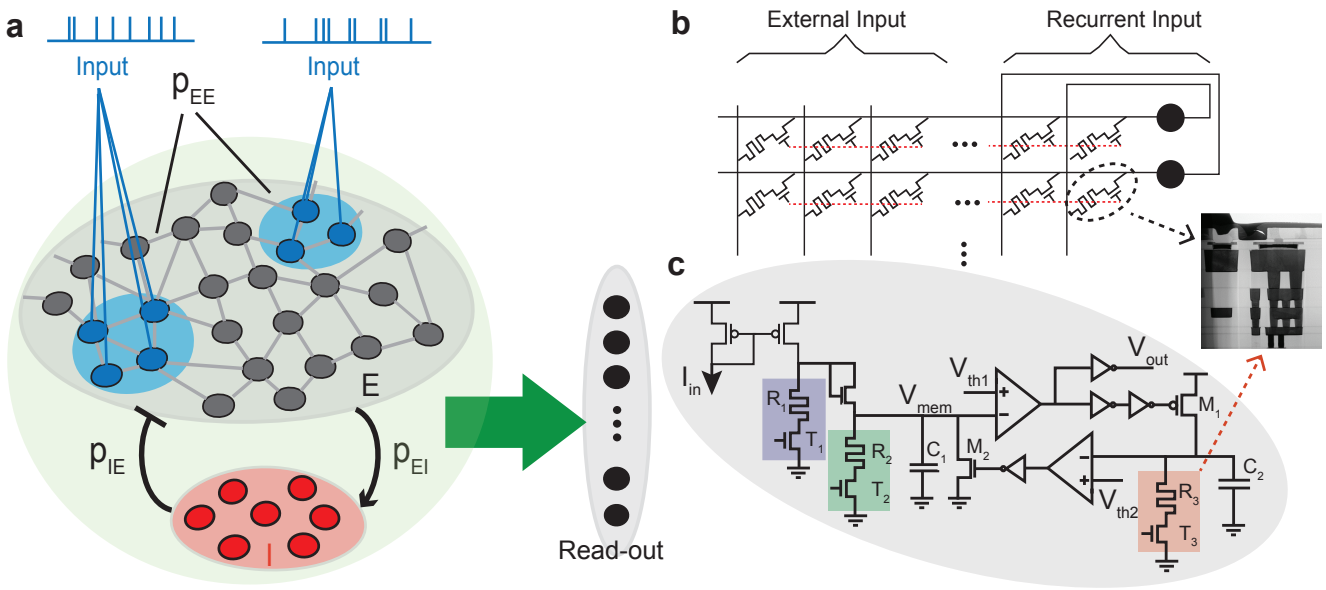

Figure 1. SRNN and its hardware implementation. (a) The SRNN is composed of a recurrent pool of excitatory neurons whose connections are formed by random fixed weights (static) or through learning (MEMSORN). The network is excited by spatio-temporal inputs activating sub-populations, shown in blue. Each of the sub-populations encodes a particular part of the sequence. The excitatory neurons are connected to an inhibitory population (shown in red), among which there are no recurrent connections. Both excitatory and inhibitory populations contribute to the activation of the output, via the readout connections (green arrow). Each neuron in the readout is assigned to a different prediction class. (b) A possible hardware implementation of the SRNN. Neuron's recurrent and external input connection are implemented by RRAM devices assembled in a crossbar array to which, inputs and neurons are connected at its columns and rows respectively. (c) Neurons are implemented using a hybrid CMOS/RRAM design. RRAMs hold the parameters of the neurons, such as gain (purple 1T-1R), leak (green 1T-1R) and refractory period (red 1T-1R).

for on-chip learning[18–21] which cannot be simply introduced in a digital implementation[6,22]. As biological networks rely on small unreliable components for reliable learning, they can provide guidance for learning with RRAM devices. Brain-inspired unsupervised Hebbian learning strategies have already been extensively explored in adaptive memristive neuromorphic architectures[13,23–25]. In these works, the RRAM conductance changes towards a more/less conductive state based on the correlation/anti-correlation between its pre- and post-synaptic neurons. However, Hebbian learning by itself cannot robustly lead to self-organization, as it implements a greedy mechanism which can lead to unstable dynamics[26]. To achieve self-organization in memristive neuromorphic architectures, a multitude of plasticity mechanisms need to be at play together, with properties and dynamics that match the physics of the underlying adaptive hardware substrate[27,28].

Here we present *MEMSORN*: a hardware architecture inspired by SORN with multi-timescale on-chip plasticity rules. MEMSORN is developed following a device-algorithm co-design approach exploiting the physics of the employed RRAM devices taking advantage of its variability. We design and fabricate the RRAM-based synapse and neurons in 130 nm Complementary Metal-Oxide-Semiconductor (CMOS) technology integrated with HfO_2-based RRAM devices. Based on the statistical measurements from these designs, we derive the local *technologically plausible* plasticity mechanisms (Hebbian and Homeostatic), and apply them in the MEMSORN architecture. We benchmark the network on a sequence learning task, and show that this approach exploits the intrinsic variability of the RRAM devices and improves the network's accuracy as a function of sequence length, learning rate, and training epochs. As a control experiment, we apply the same problem to a randomly-connected recurrent network and show that MEMSORN increases the accuracy by about 15%. This work represents a fundamental step toward the design of future *neuromorphic intelligence* devices and applications.

Results

Inspired by SORN[3], we implemented two recurrently-connected networks of Leaky Integrate and Fire (LIF) neurons: one randomly connected with fixed weights (static) and one with connections that change through learning (MEMSORN). Both networks consist of an excitatory pool of recurrently connected neurons, and an inhibitory pool of neurons that inhibit the excitatory ones, along with a read-out layer fully connected to the two pools. The inhibitory neurons balance the activity of

excitatory neurons by providing a negative feedback[29,30]. Inspired by neuro-anatomy considerations on cortical circuits, we divided the excitatory and inhibitory population to 80% and 20% of the total number of neurons, respectively[31] (see Methods). Different sub-populations of neurons are stimulated by different parts of the input sequence. In both networks, the activities of all the recurrent neurons is fed to a linear classifier at the readout which learns to distinguish between different classes of input (see Fig. 1a).

Hardware Implementation

Architecture To implement the network in hardware, we designed a crossbar memory architecture (Fig. 1b). Its rows are connected to the neurons and its columns are connected to either external inputs or to a recurrent input from another neuron. We employed RRAMs both in the design of the synapses at the cross-points holding their strength (Fig. 1b), and in the design of the neurons holding their internal parameters (Fig. 1c); Each synapse contains a transistor in series with an RRAM (aka 1T-1R), with the free side of the transistor and the RRAM connecting to the rows and the columns, respectively; Each neuron implements the LIF model shown in Fig. 1c. This hybrid CMOS/RRAM neuron design encompasses three RRAMs whose value set the neuron time constant (shown in green), gain (shown in purple) and refractory period (shown in red) (see Methods)[32]. The adaptive nature of RRAM allows for learning both the synaptic and internal neuron parameters in an on-chip and online fashion.

As soon as an input spike arrives to a column, a voltage is applied across the corresponding synaptic RRAMs, giving rise to a current, through Ohm's law. All currents are summed at the rows, and are integrated by the corresponding neurons[19]. The input to the neuron is multiplied by the gain ($R1/R2$), and is integrated on the membrane capacitance C_1 with a time constant determined by R_2C_1. As soon as the voltage on C_1 passes threshold V_{th1}, the neuron generates a voltage spike and sends it to both to V_{out} and to the feedback path. In the feedback path, the neuron's spike is integrated as C_2 voltage whose time constant is set by R_3C_2. As soon as C_2 voltage passes threshold V_{th2}, the membrane capacitance C_2 is reset and the neuron awaits the next input current.

Synapse and neuron characteristics We fabricated and measured a 4 kb synaptic crossbar array along with the hybrid CMOS/RRAM neurons, using 130 nm CMOS technology integrated with HfO_2-based RRAMs.

In the synapses, we can induce a change by applying a voltage across the RRAM devices. The device state changes from a High-Resistive State (HRS) to a Low-Resistive State (LRS) (SET operation) by applying a positive voltage between the positive and negative terminals of the 1T-1R, while applying a voltage at the gate of the transistor, V_{gate} to control the current passing through it during programming. Applying a negative voltage across the 1T-1R switches the device from LRS to HRS (RESET operation). Both SET and RESET operations produce changes in a stochastic manner. This results in a distribution over the resistance values given a programming condition[32–34]. We define a threshold at $50k\Omega$ for the resistance marking the border between HRS and LRS, and characterize the SET and RESET properties; The probability of SET operation as a function of the voltage applied to the 1T-1R cell is shown in Fig. 2a as a Cumulative Distribution Function (CDF) which follows a sigmoidal function. The RESET operation is characterized in Fig. 2b as a function of the voltage applied across the devices, with different gate voltages. The distribution of HRS values for a RESET voltages of $2V$ is shown in Fig. 2c. The distribution fits well with a log-normal function[34].

In the neurons, we measured the output firing pattern in response to a spike train as is shown in Fig. 2 d, e and f. Setting R_1 and R_2 with different values increases (Fig. 2 d) or decreases (Fig. 2 e, f) the neuron's time constant, and thus changes the likelihood of the neuron firing. In sensory-motor applications, matching the dynamics of sensory signals to those of the electronic circuit in the processing hardware can minimize the system power consumption and maximize the Signal to Noise Ratio (SNR)[4]. Therefore, to obtain neuron's time constants of millisecond range, on the order of sensory signals, while limiting the size of the capacitors (to minimize area usage), the neuron's RRAM devices should be operated in their HRS ranging from M Ω to G Ω (Fig. 1b).

Technologically-plausible algorithms

With the *technologically plausible* algorithm design, we aim to optimize the hardware implementation of algorithms by taking the hardware physics into account while developing the algorithm. Figure 3 depicts the algorithms for the two static and MEMSORN networks which are derived based on the synapse and neuron measurements of Fig. 2.

Static network The algorithm for static network (i.e., with fixed random weights) is depicted in Fig. 3b, c. The synapse and neuron behavior is fixed *a priori*; the synaptic connections are set randomly, by comparing the probability of connections in different populations to a random number, and if higher/lower, induce a SET/RESET to the devices (Fig. 3b, See Methods for details); the neuron parameters are sampled from the HRS log-normal Probability Distribution Function (PDF) derived from the measurements of Fig. 2c, equivalent to applying $V_{RESET} = 2V$ to the devices.

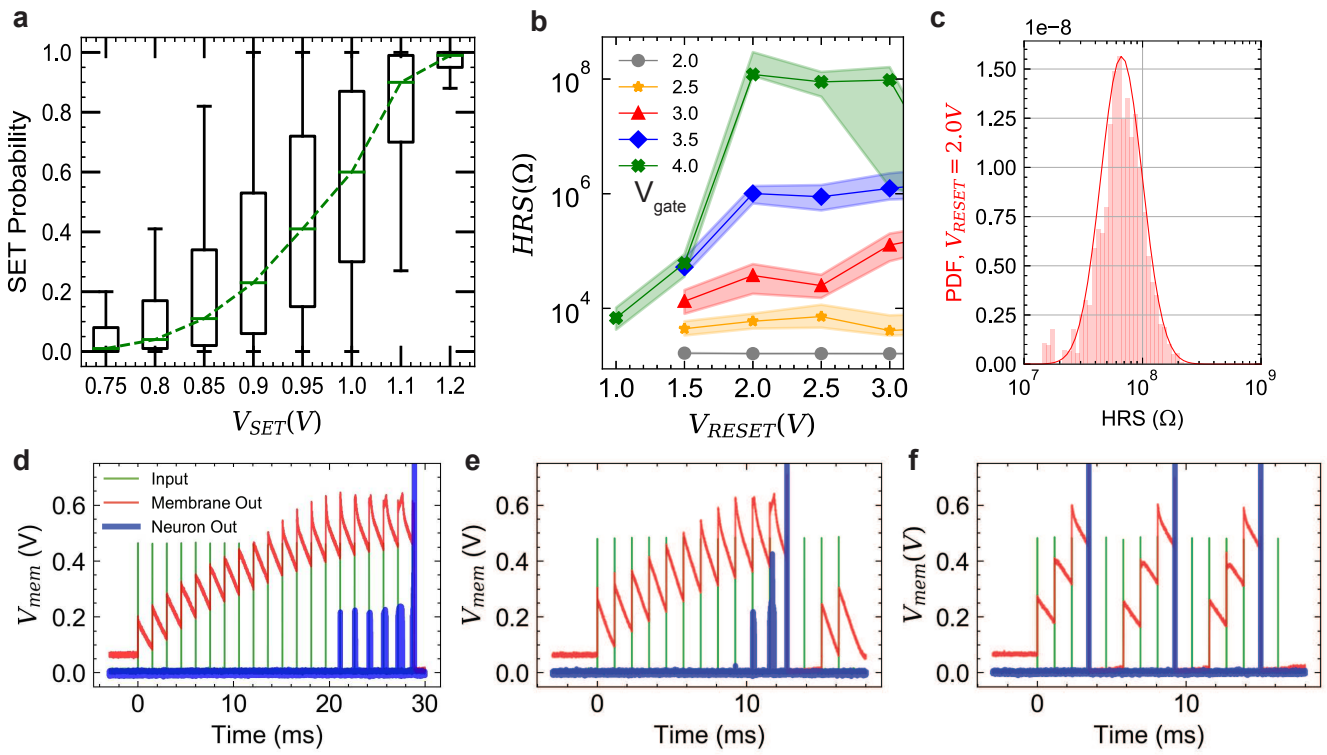

Figure 2. Measurements from fabricated synapse and neurons in 130 nm technology integrated with HfO_2-based RRAM. (a), (b), (c) Experimental results from the fabricated 4 kb synapse array. (a) SET characteristics; The box plot represents the SET probability as a function of the SET voltage, over the RRAM population; the green bar represents the median value, the box lower and upper limits represent the $\pm 25\%$ and $\pm 75\%$ quartile respectively, and the whiskers show the $\pm 95\%$ quartile. Dashed green line connecting the median values shows the emerging sigmoidal behavior of the SET probability over the SET voltage. (b) RESET characteristics; HRS measurements as a function of the RESET voltage applied across the device, for different gate voltages applied to the transistor (T) (c) The HRS distribution at $V_{RESET} = 2.0V$ and $V_{gate} = 4V$ which is well fit to a log-normal distribution, and we used in our neuron model. (d), (e), (f) Experimental results from the fabricated neuron. The neuron is excited by a train of spikes with a pulse width of $1\mu s$ and a magnitude of $450\,mV$ and a frequency of $1\,kHz$ (green). Changing the current through R_1 and R_2 paths changes the gain and time constant of the neuron. (d) Time constant of the neuron is set to $\approx 1.5\,ms$ by controlling the leak out of C1. Gain is set so as to make neuron integrate for many pulses before it fires. (e) Current through R1 path is decreased to increase the neuron's gain which makes the neuron fire faster. (f) Current through R2 and thus neuron's time constant is decreased to $\approx 1\,ms$, making the neuron fire more frequently.

MEMSORN The *MEMSORN* plastic network *self-organizes* to form multiple cell-assemblies. This is done through changing the RRAMs in the synapse and neuron parameters through learning. The excitatory synapses undergo a Hebbian-type plasticity, Spike Driven Synaptic Plasticity (SDSP), which changes the synaptic RRAM based on the correlation between the input (pre-synaptic) and output (post-synaptic) neural activities[7,35]. In addition, the neuron parameters undergo Homeostatic plasticity, Intrinsic Plasticity (IP), which acts as regulatory mechanism to keeps the neuron's firing activity within a desired range[36]. Both forms of plasticity are well suited for the implementation on CMOS and RRAM hardware.

Following the SDSP rule, the RRAM resistance of a synapse is decreased/increased, on the onset of its pre-synaptic spike, if the membrane potential of the post-synaptic neuron is higher/lower than V_θ threshold (Fig. 3d).

On the other hand, IP changes the neuron's RRAM to maintain it's output firing rate, f_n, close to a target firing frequency, f_T, within a tolerance of σ (Fig. 3e). If f_n lies outside of these boundaries, the RRAMs in their HRS are updated accordingly. For simplicity, we have chosen to only update R_2 which simultaneously changes both the gain and the time constant of the neuron. Changing the gain will additionally implement synaptic scaling which is another homeostatic plasticity mechanism, used in conjunction with IP in the original SORN paper[3]. To tune R_2 in HRS, it is first SET and then RESET. SET is done probabilistically proportional to the difference between f_n and f_T (δ). Once SET, The RESET operation with a fixed V_{RESET} effectively samples a new HRS value from a log-normal PDF. Therefore, neurons with a frequency deviating significantly from the target will change their leak and gain by acting on R_2, to adapt their firing rate. Note that since the amplitude of V_{RESET} is fixed, the sampled HRS value is drawn from a single distribution, which makes the search for the correct resistance

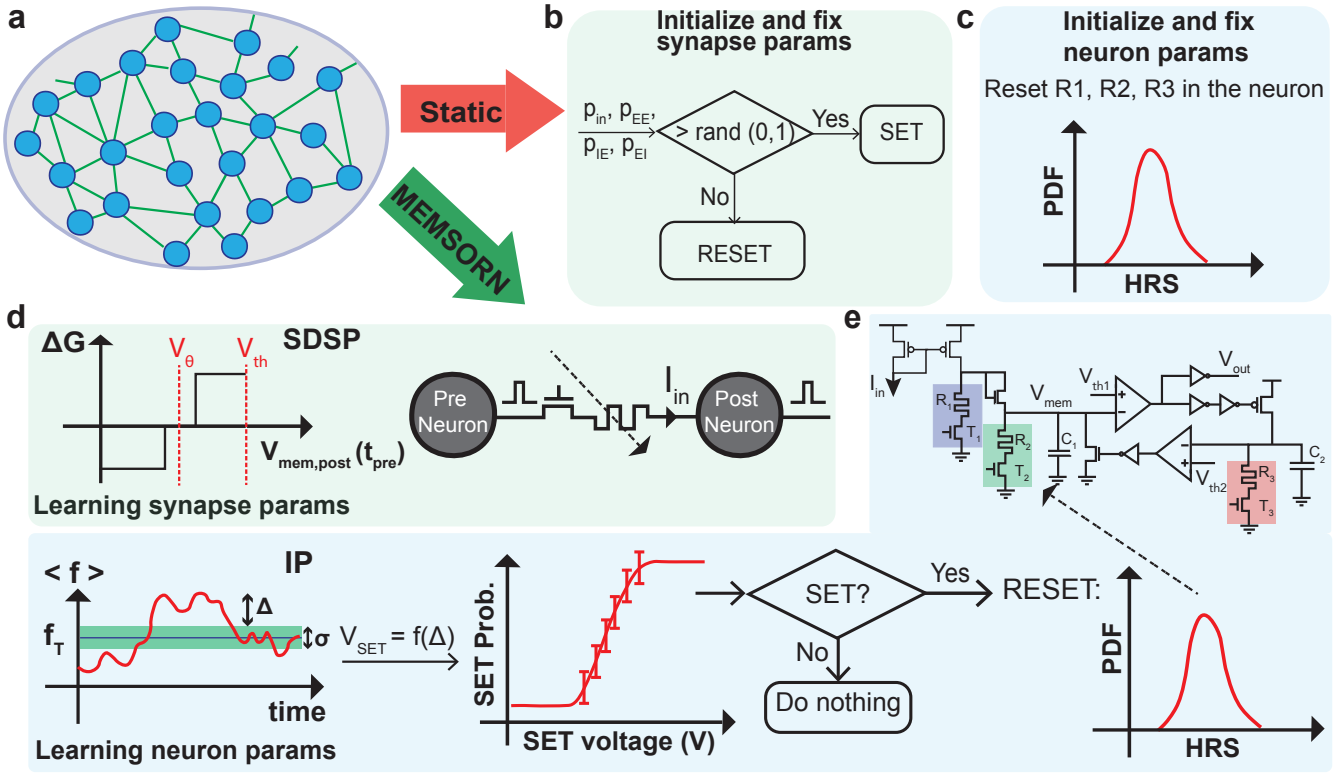

Figure 3. Technologically plausible algorithms for static and MEMSORN networks. (a) An SRNN with excitatory connections. (b, c) Synapse and neuron parameters for static network. Both values are fixed after an initialization process. (b) Synaptic parameters are initialized based on comparing the probability of different connections with a random number. (c) Neuronal parameters are initialized by resetting the resistors R_1, R_2 and R_3 which is equivalent to sampling from a log-normal distribution around a mean resistance that is a function of the reset voltage. (d, e) Synaptic and neuron parameters for MEMSORN network. Both parameters are learned throughout the input sequence presentation. (d) Synaptic parameters are learned based on the Hebbian-based SDSP learning rule. At the time of the pre-synaptic event (t_{pre}), weight (conductance) of the synapses are increased/decreased, if the membrane potential of the post-synaptic neuron ($V_{mem,post}$) is higher/lower than V_θ. (e) Neuron parameters are changed based on the IP algorithm which tries to keep the firing rate of each neuron ($< f >$) in a healthy regime ($f_T \pm \sigma/2$). If the neuron's firing rate goes beyond this regime, neurons' R_2 is first SET probabilistically and then RESET. The RESET process samples a new value for R_2 from the log-normal distribution of HRS values.

values *non-guided*. This reduces circuitry overhead with respect to an alternative algorithm in which the RESET operation is performed by adapting the V_{RESET} to the deviation of the f_n from f_T (i.e. $V_{RESET} \propto |f_n - f_T|$)[36] (see Methods).

Benchmark

To validate our approach, we used the same benchmark proposed in the original SORN paper[3]: a sequence learning counting task for predicting the next sequence at the output. The network receives a shuffled alternation of two input sequences of length $n+2$ of 6 possible characters in $[A,B,C,D,E,F]$. In both sequences, either character B or E are repeated n times (Fig. 4a). Examples of these sequences are $S_{1,n} : [A,B_1,B_2,...,B_n,C]$ and $S_{2,n} : [D,E_1,E_2,...,E_n,F]$. The goal is to learn to predict the next character given all the previously-presented ones, *i.e* $P(next-character_i|\sum_j^{i-1} shown-character_j)$. After fixing the length of the sequence, the network has to learn to count the repetition of characters B and E by means of a reliable dynamical state.

We applied the counting task to the static and MEMSORN networks and compared their performance. The network is asked to differentiate between $n = 10$ repetitions of the same symbol, presented in the middle of the two sequences. Each symbol's position in the two sequence is assigned to one output neuron in the readout whose activity represents the network's prediction. Fig. 4b shows the most active output neuron indicating the prediction, compared to the expected output neuron activity in both networks (output 1-12 for $S_{1,n}$ and 14-25 for $S_{2,n}$).

The static network is capable of separating only the first repetitions, whereas the MEMSORN network can successfully resolve all the repetitions, forming a staircase of activity in the output neurons along time, matching the output to the target. Since the two sequences are randomly alternated, the output of the network under the presentation of the last symbol in the sequence cannot be predicted. This applies to outputs 0, 13 and 25 in Fig. 4b. The internal dynamics in the static network

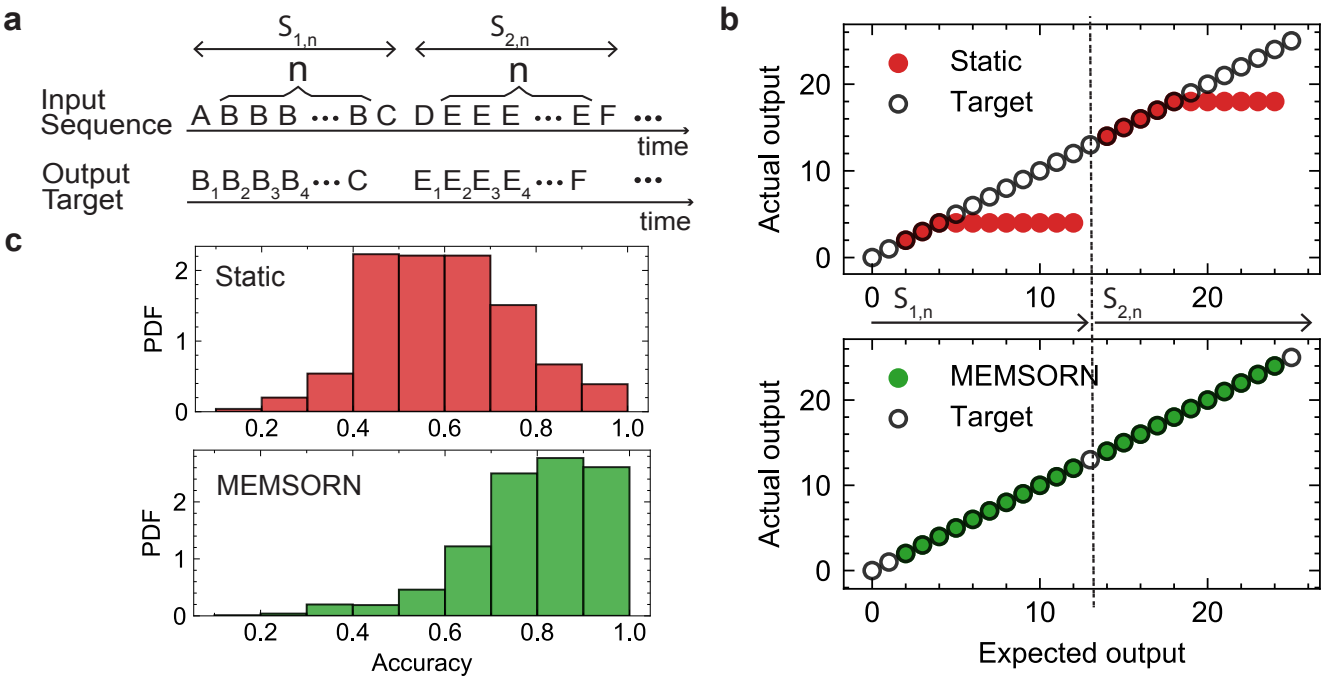

Figure 4. Static and MEMSORN performance comparison. (a) Sequence learning task. Two input sequences of $S_{1n} = ABB...BC$ and $S_{2n} = DEE...EF$, where Bs and E are presented n times, are fed to the network. Each letter represents part of the sequence. The task is a counting task where the output should predict the next sequence letter, for which it needs to keep count on the number of Bs and Es that have been presented. (b) Output prediction in static and MEMSORN compared to the expected output. Each output neuron is assigned to a certain position in the two sequences. The first 12 neurons are assigned to $S_{1,n}$ and the second 12 are assigned to $S_{2,n}$. The static network is capable of separating only the first repetitions, whereas the MEMSORN network can successfully resolve all the repetitions, forming a staircase of activity in the output neurons. The missing output in MEMSORN is due to the uncertainty about the next sequence as it is randomly chosen. (c) Histogram of the accuracy in the static (red) and MEMSORN (green) networks tested on 1000 different networks, for the counting task with sequence length (n) of 10. The mean of the accuracy distribution in MEMSORN network moves towards higher values of accuracy (mean of 0.756 for MEMSORN network compared to 0.596 in the static network). Also, the number of low-accuracy networks in MEMSORN is greatly reduced compared to the static network (about four times).

saturates and lands on an attractor state from which no further information can be extracted. The MEMSORN network, instead, is capable of forming more complex dynamics that allow for fading memory to form and separate the repetitions in the input sequence. Figure 4c illustrates the histogram of the accuracy calculated over 1000 networks initialized differently for both networks on the counting task with the sequence length of $n = 10$. As shown, the mean accuracy of the MEMSORN network is shifted to the higher accuracy levels compared to the static network. (Mean accuracy of 0.756 compared to 0.596 respectively). The standard deviation is due to the random initialization of the connections and the variability of RRAMs, implementing both the weights of the connections and the parameters of the neurons (See Methods.) Taking into account the hardware constraints, our statistical analysis shows that by enabling learning inside the recurrent network, there is a higher probability of obtaining a more accurate network; i.e. the number of learned networks that can correctly predict the next letter with an accuracy of more than 0.8, is four times that of the static network.

Analysis on the effect of variability in MEMSORN

RRAM devices undergo cycle-to-cycle and device-to-device variability as was confirmed with our measurements in Fig. 2. To understand the effect of variability in MEMSORN, we performed simulations on four cases: (i) No device variability and IP operation off; (ii) Variability in devices receiving the SDSP rule, and IP operation off; (iii) Variability in devices receiving the SDSP, and IP operation on without variability; (iv) Variability in both SDSP and IP learning with standard deviation for the IP devices set as 0.1, taken from our measurements. It is worth noting that in condition (iii), since there is no variability in IP operation, the same initial value is always applied when IP is acting.

Figure 5 shows the network performance under these four conditions. The figure demonstrates the positive effect of regularizing IP mechanism, and how MEMSORN network exploits the different sources of variability of the RRAM devices to increase its accuracy on the sequence learning task; Figure 5a plots the histograms of accuracy for every 100 samples of

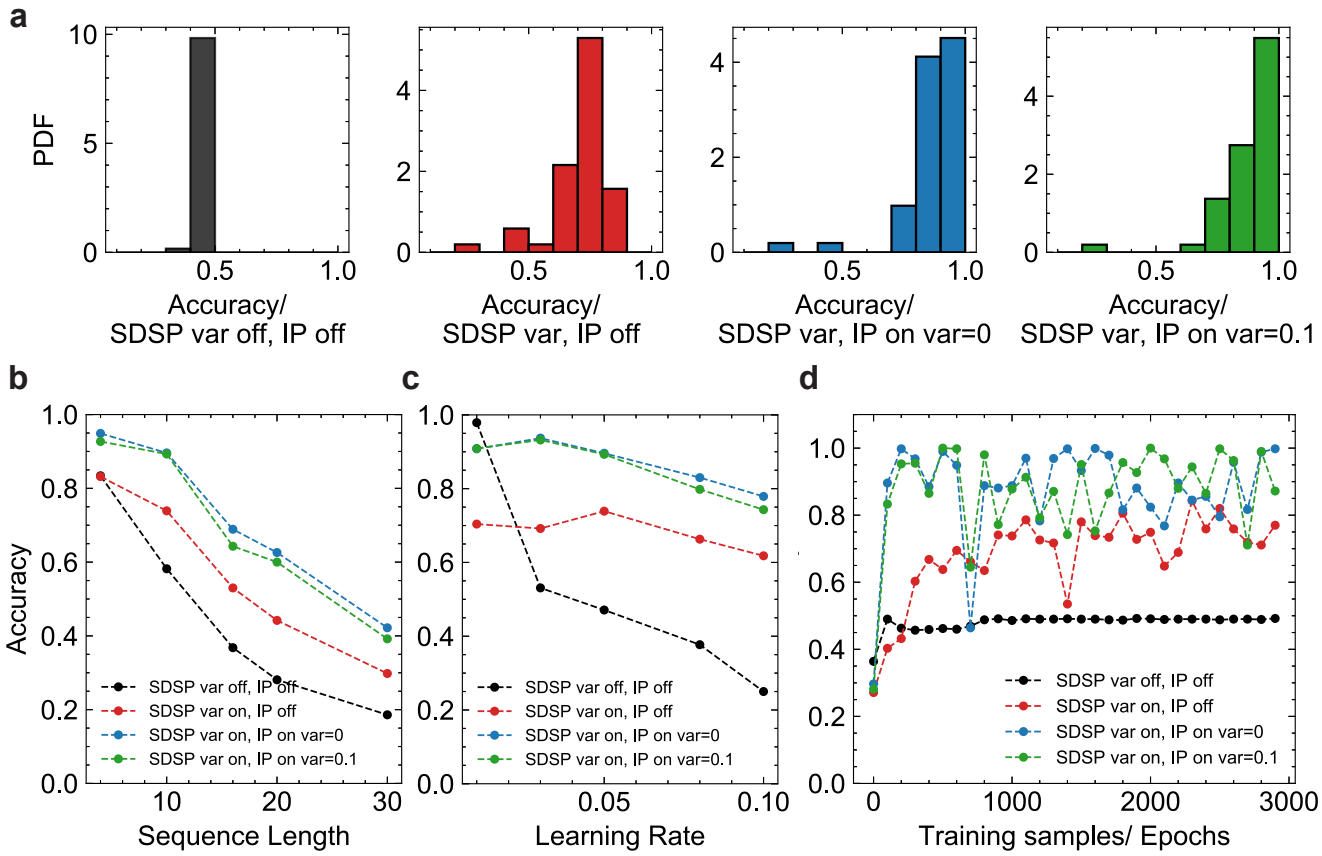

Figure 5. Performance of our proposed self-organized network (MEMSORN) under four different cases of variability in the device models: (i) No device variability and IP operation off (black), (ii) Variability in devices receiving the SDSP, and IP operation off (red), (iii) Variability in devices receiving the SDSP, and IP operation on without variability(Blue) and (iv) Variability in both SDSP and IP learning with standard deviation for the IP devices set to 0.1, taken from our measurements (green). (a) Histogram of accuracy for 500 networks confirms the higher accuracy for the networks including variability and IP compared to other conditions. (b) Accuracy of the MEMSORN network on the counting task with respect to the sequence length. As the sequence length increases, the network needs to remember increasingly more input symbols in the form of a dynamical state of the network, and thus the accuracy drops. Introducing variability, calibrated on measured data, helps the accuracy of the network as all the cases with variability outperform the case without any variability. (c) Average accuracy of the MEMSORN network (for every 100 samples between 1000 to 5000 training epochs) on the counting task with length of 10 as a function of different learning rates. Introducing variability makes the network robust to hyper-parameter change. (d) Learning evolution of the network accuracy on the counting task with sequence length of 10 for the four variability cases. Condition (i) has much less noise, but has an overall lower accuracy (less than 40%) than the cases where variability and IP are introduced.

learning in the MEMSORN network for all the variability conditions. The histograms show that introducing IP operation, and any source of variability shifts the mean of the accuracy of the network to higher values; Figure 5b illustrates the accuracy as a function of the sequence length. As the sequence length increases, the network needs to remember increasingly longer sequences which tests its fading memory[37]. Thus, the accuracy of the network drops with longer sequences. It is worth noting that as the sequence length increases, the number of output neurons increases, and thus the baseline chance level accuracy reduces. Figure 5b confirms that the networks including IP and added source of variability outperform other conditions. Figure 5c depicts the network accuracy as a function of the SDSP learning rate (See Methods). Despite that large learning rate results in a consistent drop of accuracy, introducing variability suppresses accuracy degradation. This suggests that the noise introduced by the variability of the RRAM devices is beneficial for the stability of the network making them less sensitive to hyper-parameters and low bit resolution. This is because through learning with noise, the algorithm finds a set of parameters that are more insensitive to noise. Figure 5d shows the accuracy evolution of the MEMSORN network during learning epochs. Each epoch consists of presenting one of the two sequences which are presented to the network with a random order. Condition (i) without any variability and IP operation(black) leads to a more stable learning dynamics but also lower performance. Instead, adding noise to SDSP or adding the IP operation causes some instability in the network, but also allows for much higher overall accuracy. Finally, combining the variability in SDSP with that of IP leads to the best performance compared to other conditions.

The positive effect of variability is because a distribution of parameters due to variability provides a larger space of parameters for search during learning which helps the network to explore and reach a better set of parameters for the task.

Clustering analysis

To understand the dynamics of the static and MEMSORN networks, we performed clustering analysis on the firing activities of the neurons inside the excitatory pool. Figure 6 shows the result of the clustering analysis. First, we reduced the dimensionality of the neural activity using Principle Component Analysis (PCA) (see Methods). Figure 6a plots the PCA of both network activities in response to 50 sequences of length 10. Temporally adjacent letters in the sequence line up next to each other in the principle component space. This indicates the higher structural richness in MEMSORN compared to the static network. Moreover, this helps with the classification accuracy in the readout layer, since the sequences become more linearly separable as indicated by the PCA plot. Figure 6b plots the histogram of explained variance in the firing rate of the random and MEMSORN networks with respect to the first 20 principle components. The explained variance is about 11% more in MEMSORN network compared to the random network suggesting more orderly dynamics in MEMSORN network.

Additionally, we performed a hierarchical clustering analysis on the activity of the SRNN which reveals the formed cell-assemblies (see Methods). The result is indicated by the dendogram in Fig. 6c showing an increase in the number of uncorrelated clusters in MEMSORN network compared to the random network. This is as a result of more structure emerging from the learning in the recurrent network which is in agreement with the unsupervised memory formation in cell-assemblies as we argued in the introduction.

Discussion

Following a device-algorithm co-design approach, we presented MEMSORN, a hardware architecture that uses its intrinsic properties to self-organize and learn a sequence prediction task. We used hybrid CMOS/RRAM technology as our hardware substrate, and presented experimental results from the implementation of neurons and synapses in this technology. We then used these measurements to derive "technologically plausible" local learning rules which gives rise to self-organization in a SRNN. The self-organization proved to improve the accuracy of the SRNN compared to a fully random one, by more than 15% on a sequence prediction task.

The unique property of RRAM, compared to other technologies, which was highlighted in this work, is its intrinsic variability. The variability provides a distribution of analog values which equips the learning with a large parameter space. We showed that this variability improved the sequence learning by about 30% for different sequence lengths. Moreover, introducing the variability while learning, de-sensitized the network to the hyper-parameters, specifically the learning rate. This variability is a physical property of the RRAM and is present for 'free'. Implementing such randomness using digital circuitry requires bulky circuits such as Linear-feedback shift registers, which use and calculate an algorithm to generate pseudo-random numbers. Therefore, our approach of "technological plausibility" paves the way for building systems that are potentially more area and also power efficient, as their physical structure gives rise to their function.

Technologically plausible co-design approach closes the gap between the ideas inspired by neuroscience and their applications, algorithms, circuits and devices. Taking the physics of the devcies into account, we designed algorithms that match and exploit them, while designing circuits that implement the algorithms interfacing with the devices that can be used to solve a real-world problem.

We envision a future where intelligent chips enabled by MEMSORN-like designs receive sensory information from the environment and self-organize themselves to interact smoothly with it. This work is a step toward that direction.

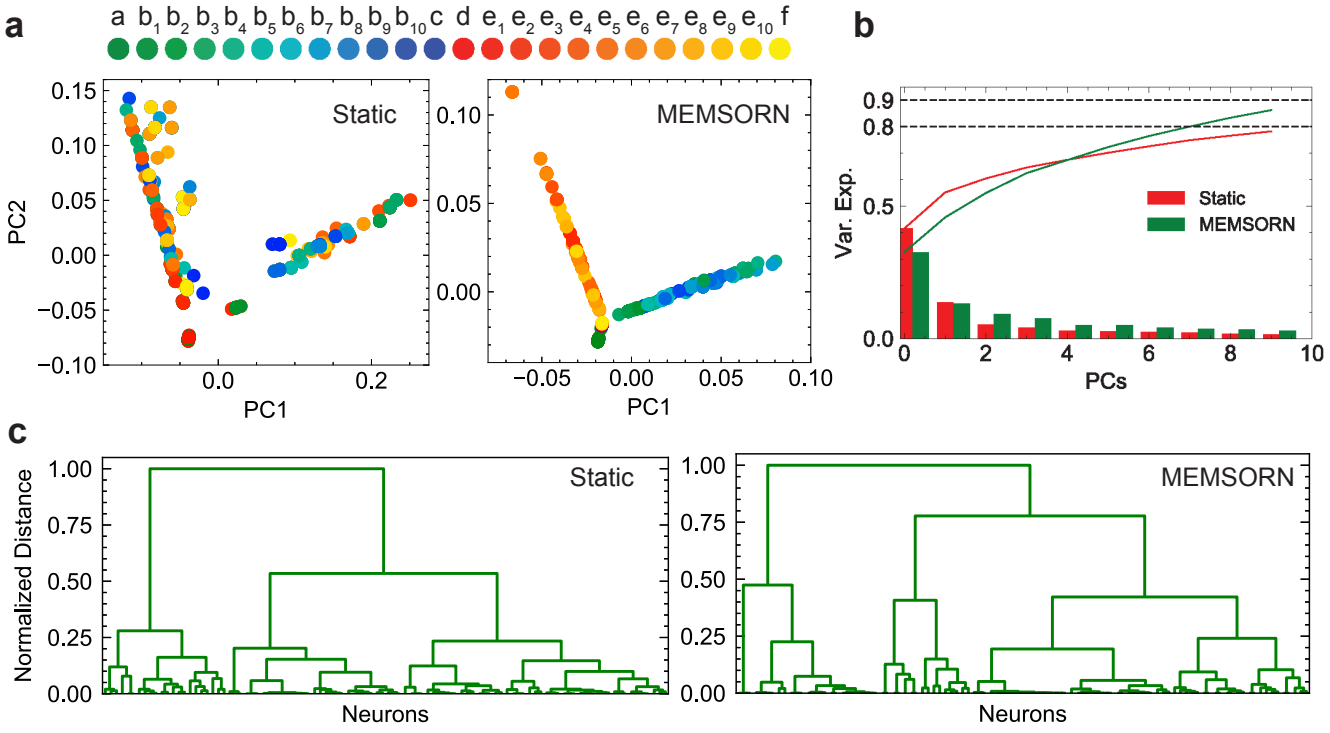

Figure 6. Clustering analysis on the spiking activity of the networks for static and MEMSORN architectures on the counting task with sequence length of 10 . (a) Principle Component Analysis (PCA) applied to the firing rate of the two networks in response to 50 sequences of length 10 (600 letters). Each color is assigned to a different position of the letter in the sequence with similar colors encoding temporal adjacency of the letters in the sequence. In Principle Component (PC) space, the different input conditions form random clusters in static network that are not well separated. On the other hand, in MEMSORN network compact clusters are formed which are well separated for different input conditions. (b) Histogram of captured variance by the first 20 PCs in the static and MEMSORN networks. The explained variance amounts to 79% for static network compared to 87% in the MEMSORN one suggesting more orderly dynamics in MEMSORN network. (c) Dendogram of static and MEMSORN networks showing the hierarchical relationship between clusters of neurons. The normalized height of the dendogram indicates the distance between the clusters and the links indicate the order in which the clusters are joined. For any given distance, the number of branch numbers for MEMSORN are larger than those for the static network, indicating that the clusters in MEMSORN are better-structured.

Methods

Fabrication/integration

The circuits of Fig. 1 have been taped-out in 130 nm technology at CEA-Leti, in a 200 mm production line. The Front End of the Line, up to metal 4, has been realized by ST-Microelectronics, while from Metal 5 upwards, including the deposition of the composites for RRAM devices, the process has been completed by CEA Leti. RRAM devices are composed of a 5 nm thick HfO_2 layer sandwiched by two 5 nm thick TiN electrodes, forming an $TiN/HfO_2/Ti/TiN$ stack. Each device is accessed by a transistor composing the 1T-1R unit cell. The size of the access transistor is $650\,nm$ in width. 1T-1R cells are integrated with CMOS-based circuits by stacking the RRAM cells on the higher metal layers.

Device Measurements

For programming and reading the RRAM devices, Source Measure Units (SMU)s from the 4200 SCS Keithley machine were used. We performed statistical analysis from the switching characteristics of a 4 kb array of HfO_2-based RRAM. A SET/RESET operation is performed by applying a positive/negative pulse across the device which forms/disrupts a conductive filament in the memory cell, thus decreasing/increasing its resistance. When the filament is formed, the cell is in the LRS, otherwise the cell is is the HRS. For a SET operation, the bottom of the 1T1R structure (columns in Fig. 1b) is conventionally left at ground level, and a positive voltage is applied to the 1T1R top electrode (rows in Fig. 1b). The reverse is applied in the RESET operation. Typical values for the SET operation are V_{gate} in $[0.9-1.3]\,V$, while the V_{top} peak voltage is normally at $2.0\,V$. For the RESET operation, the gate voltage is instead in the $[2.75, 3.25]V$ range, while the bottom electrode is reaching a peak at $3.0\,V$. The reading operation is performed by limiting the V_{top} voltage to $0.3\,V$, a value that avoids read disturbances, while opening the gate voltage at $4.5V$.

SET and RESET statistics To ensure the resistive switching is *deterministic*, i.e., the device definitely makes the transition from HRS to LRS, a strong V_{SET} is usually applied across the device. If the programming voltage is lowered, a sub-threshold SET is obtained, which makes the switching operation *stochastic*. We analyzed the sub-threshold SET operation on a population of 4096 1 transistor- 1 RRAM (1T-1R) devices by applying a different range of V_{SET} voltages for 100 cycles, while setting the gate of the transistor to $1.7\,V$. To perform a RESET operation, an opposite voltage V_{RESET} with respect to the SET operation is applied and the gate is biased with V_{gate}. HRS of the devices after the RESET operation for different V_{RESET} and gate voltages V_{gate} were recorded over 100 cycles. Based on these measurements, we derived the statistical model for the stochastic SET and RESET which were used in our simulations.

Technologically plausible self-organizing network

Static network The static fully-random network is similar to the Liquid State Machine (LSM)[38] where two populations of excitatory (E) and inhibitory (I) neurons are randomly connected. The excitatory/inhibitory neurons increase/decrease the post-synaptic potential of the neurons to which they are connected. The SRNN neuron population is divided into 80% of excitatory and 20% of inhibitory neurons[38]. An input layer encodes the input information by means of Poisson spike trains. The choice of a noisy input is common practice for Spiking Neural Networks (SNNs) and is justified by the noise-resilient nature of SNNs and stimulation of plasticity mechanisms. The input is randomly projected to the SRNN with a probability of p_{in}, set at 0.2 in this work. The excitatory population connects to itself with a probability of p_{EE}, and connects to the inhibitory population with the probability p_{EI}. The inhibitory population connects to the excitatory population with probability p_{EI}. The SRNN represents a complex dynamical system governed by many parameters. In detail, neurons have three parameters (gain, time-constant and threshold), and synapses have 2 parameters (time constant and weight). Table 1 shows the parameter values for the initialization of the SRNN. The output from each of the neurons in the excitatory pool is projected to the output layer, constituted of neurons that encode the output of the network.

MEMSORN *MEMSORN* is equipped with the two technologically plausible learning rules of SDSP and IP. These local learning rule are only applied to the excitatory neurons and EE connections.

Modified SDSP The measure of correlation in SDSP is the difference between the membrane potential of the post-synaptic neuron V_{mem} to a defined threshold, V_θ at the time of the pre-synaptic spike t_{pre}. The weight update on t_{pre} is defined as:

$$w_{EE} = \begin{cases} w_{EE} + LR, & \text{if } V_{mem} \geq V_\theta \\ w_{EE} - LR, & \text{otherwise} \end{cases}$$

Where w_{EE} is the weight between the excitatory neurons, and LR is the learning rate. The SDSP rule is thus controlled by two parameters, the thresholds applied to the post-synaptic neuron membrane voltage (V_θ), and the synaptic weight increment (LR). These values are reported in table 1.

Intrinsic Plasticity In SRNNs equipped with plasticity mechanisms, Hebbian synaptic plasticity is thought to create clusters of tightly-bonded neurons, known as attractors. In these networks IP controls the growth of such attractors and in turn limits the dynamics of neural microcircuits. This effectively improves the information transfer across the SRNN and eventually to the output.

Algorithm 1: IP algorithm

Initialization: $R = RESET_{init}(V_{Reset})$
while $t < taskDuration$ **do**
  for *Neurons in the excitatory pool* **do**
    if $@t_{post}$: $|f_{neuron} - f_T| > \sigma/2$ **then**
      # Sub-threshold Stochastic SET
      $V_{set} = f(|f_{neuron} - f_T|)$
      $p_{set} = P\_subthSET(V_{Set})$
      if $R_{final} < 50k\Omega$ **then**
        # RESET
        $R_{HRS} = RESET()$

Technologically-Plausible IP Algorithm 1 describes the technologically plausible IP algorithm to change the conductance of RRAMs in order to keep the firing rate in a healthy regime. A target firing frequency f_T with an error margin σ is defined as the desired range, and the neuron measures its firing rate f_n with respect to the boundaries $f_T \pm \sigma/2$. If f_n moves outside of these boundaries, the value of HRS needs to be updated. To do so, the RRAM is SET with a subtheshold SET voltage which is proportional to the difference between the target and neuron activity. The higher the difference, the higher the SET voltage and thus the higher the probability of setting the device. If the device is SET (i.e., the final resistance $R_{final} < 50k\Omega$), we then RESET the device to sample from its internal distribution and find a new value that sets the time constant and gain of the neuron. IP rule is thus controlled by three parameters of up and down thresholds applied to the neuron's firing rate, and the RESET voltage). The values that are used for all the variables in the learning algorithms are in table 1.

Neurons			Synapses		SRNN	
	Excitatory	Inhibitory				
Number of neurons	160	40	τ	1 ms	p_{EE}	2%
R_2	trained by IP	$1\,G\Omega$	Weight	(trained by SDSP)	p_{II}	0%
R_1	$400\,M\Omega$	$600\,M\Omega$			p_{EI}	2%
C_1	10 pF	10 pF			p_{IE}	10%
τ_{Ca}	100 ms	100 ms				
V_{th}	0.2 V	0.2 V				
R_3	$1\,G\Omega$	$1\,G\Omega$				
C_2	2 pF	2 pF				
IP			SDSP			
F_T	50Hz		V_{th}	0.2 V		
σ	15Hz		V_θ	0.1 V		
V_{RESET}	2 V		$LearningRate$	0.01-0.1		

Table 1. Parameter values for the initialization of the SRNN. Such values are defined with small-to-absent tuning, with the only aim to guarantee a minimal activation of the network, so to fully rely on the plasticity mechanisms (SDSP and IP) to improve performance. Some parameters, such as the magnitude of RRAM resistance in HRS and the Membrane Capacitance, are forced by technological constraints.

Sequence presentation and learning

Two patterns of S_{1n} and S_{2n} are presented to the network in random order and sequentially. Each symbol is assigned to a sub-population in the excitatory pool and upon presenting the letter, the corresponding sub-population is stimulated with Poisson spike trains with frequency of 1 kHz. Each symbol is presented for 50 ms. Once each sequence is completely presented,

we wait for 200 ms before presenting the next sequence. This will ensure the activity of the network is decayed away before the next sequence is presented.

Static network The values for the neuron and synapses are initialized and fixed at the beginning of the simulation as is shown in Fig. 3b and c. The neuron parameters are sampled from the HRS distribution fixed on $V_{RESET} = 2V$ and the synapse parameters are generated randomly based on p_{in}, p_{EE}, p_{EI} and p_{IE}. The parameters of the static network follow the values in table 1. These hyper-parameters are manually tuned to put the SRNN in an optimal condition. A general optimum operational condition for SRNNs is the *Edge of Chaos*, in which the dynamics of the network is neither chaotic nor deterministic. Traditional approaches to achieve this optimal conditions are either exploiting genetic algorithms[39] or manual hyper-parameter tuning. A random initialization of the weights in the SRNN does not guarantee the formation of multiple clusters of connected neurons, proved to be crucial for memory formation in recurrent networks[2].

MEMSORN Training of the MEMSORN is performed in two steps: first, a purely unsupervised phase in which the network is exposed to inputs and the two technologically-plausible rules of SDSP and IP shape the activity and response of the SRNN; second, the recurrent weights and neurons' RRAM states are frozen and the output is trained with logistic regression.
In the first phase, the input patterns are presented and synapses modify their weights according to the SDSP rule, creating clusters of neurons that respond to particular spatio-temporal input sequences. The input signal is converted in to the activation of different groups of neurons inside the recurrent network. In turn, this sub-groups of neuron are connected to the rest of the SRNN in random and sparse manner. This results in each input exciting the SRNN in a different way; SDSP is thus capable of exploiting these correlations between parts of the SRNN to reinforce certain connections and form different clusters in the network. At the same time, IP adapts the excitability of neurons in order to control their activity. This prevents strong clusters to take over the SRNN and form a single big group of hyperactive or inactive neurons. Therefore, the benefit of IP is to steer away the SRNN from a regime in which either most neurons are not present in information processing, or they have high output frequency and thus high energy consumption. In MEMSORN, the hyper-parameter tuning of the static network is substituted with the described unsupervised phase. The plasticity mechanisms are exploited in order to find a good enough set of parameters to optimize the performance of the SRNN, alleviating the need to carefully tune or learn the network parameters. These plasticity mechanisms are capable of finding a suitable SRNN configuration depending on the presented input which tunes the network accordingly.
Once the SRNN is tuned, the second phase begins. The activity of the neurons in the network is low-pass filtered through the calcium dynamics of the neuron circuit (τ_{Ca}), indicative of the running average activity. This running firing rate is registered at the end of the sequence and utilized in a logistic regression algorithm to calculate the output weights in the readout. The logistic regression could also be replaced with the online delta rule in an always-on fashion[40]. Such circuit implementation of the delta rule algorithm allows to train the output layer using Stochastic Gradient Descent (SGD) in a one-layer SNN. This kind of system will continuously adapt to the input stimuli in a low power always-on manner.

Clustering Analysis

We show 50 sequences of length 12 ($n = 10$, 600 letters) to the static and MEMSORN networks and record the spike times of all the excitatory population for each shown symbol. This spiking data is then low pass filtered with a time constant of $\tau = 5ms$. We take the data point at the last time step of the symbol presentation for all the symbols in the two sequence during 50 data presentations.

PCA We reduce the dimension of the SRNN activity from 160 excitatory neurons to the first two Principle Components (PCs) and plot the first and second PCs in time. This is plotted in Fig. 6a.

Hierarchical Clustering We also analyze the obtained SRNN activity using hierarchical clustering analysis. This is a type of unsupervised learning algorithm used to cluster the data points with similar characteristics. We use the agglomerative hierarchical clustering which is a "bottom-up" approach where each data point starts in its own cluster. Moving up the hierarchy, clusters are formed by joining the two closest data points resulting in increasingly less clusters with higher distance or dissimilarity. This hierarchy of clusters is represented as a tree (or dendrogram). The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample. For performing this analysis, we have used the "cluster.hierarchy.linkage" function from the python scipy library. We have calculated the distance between the clusters with the "ward" method which minimizes the total within-cluster variance. At each step, this method merges pairs of clusters which lead to a minimum increase in the total within-cluster variance after merging. This increase is a weighted squared distance between cluster centers.

Data availability

All measured data are freely available upon request.

Code availability

All software programs used in the presentation of the Article are freely available upon request.

Acknowledgements

This work is supported by Toshiba corporation and EU Horizon 2020 Memscales project (871371). We are grateful to Stefano Brivio from CNR, Italy, and Christian Tetzalff from Univerity of Gottingen, Germany for their critical comments on the manuscript.

Author contributions statement

M.P. proposed the idea. M.P, F.M and T.D conceived the experiment(s). F.M. and K.N. conducted the software experiment(s). M.P, F.M, K.N, Y.N and G.I analyzed the data. M.P and T.D designed the circuits. The circuit were fabricated at CEA-LETI under the supervision of E.V. F.M measured the fabricated chips under the supervision of E.V. M.P, F.M and G.I wrote the manuscript. All authors reviewed the manuscript.

References

1. Herpich, J. & Tetzlaff, C. Principles underlying the input-dependent formation and organization of memories. *Netw. Neurosci.* **3**, 606–634 (2019).
2. Tetzlaff, C., Dasgupta, S., Kulvicius, T. & Wörgötter, F. The use of hebbian cell assemblies for nonlinear computation. *Sci. reports* **5**, 12866 (2015).
3. Lazar, A., Pipa, G. & Triesch, J. SORN: a self-organizing recurrent neural network. *Front. computational neuroscience* **3**, 23 (2009).
4. Indiveri, G. & Sandamirskaya, Y. The importance of space and time for signal processing in neuromorphic agents. *IEEE Signal Process. Mag.* **36**, 16–28, DOI: 10.1109/MSP.2019.2928376 (2019).
5. Kreiser, R., Waibel, G., Armengol, N., Renner, A. & Sandamirskaya, Y. Error estimation and correction in a spiking neural network for map formation in neuromorphic hardware. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 6134–6140 (IEEE, 2020).
6. Davies, M. *et al.* Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* **38**, 82–99 (2018).
7. Qiao, N. *et al.* A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Front. neuroscience* **9**, 141 (2015).
8. Michaelis, C., Lehr, A. B. & Tetzlaff, C. Robust trajectory generation for robotic control on the neuromorphic research chip loihi. *Front. neurorobotics* **14** (2020).
9. Ambrogio, S. *et al.* Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature* **558**, 60–67, DOI: 10.1038/s41586-018-0180-5 (2018).
10. Li, C. *et al.* Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. *Nat. communications* **9**, 1–8 (2018).
11. Li, C. *et al.* Long short-term memory networks in memristor crossbar arrays. *Nat. Mach. Intell.* **1**, 49–57 (2019).
12. Wang, Z. *et al.* In situ training of feed-forward and recurrent convolutional memristor networks. *Nat. Mach. Intell.* **1**, 434–442 (2019).
13. Prezioso, M. *et al.* Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* **521**, 61–64, DOI: 10.1038/nature14441 (2015).
14. Dalgaty, T. *et al.* In situ learning using intrinsic memristor variability via Markov chain Monte Carlo sampling. *Nat. Electron.* **4**, 151–161 (2021).
15. Cai, F. *et al.* Power-efficient combinatorial optimization using intrinsic noise in memristor hopfield neural networks. *Nat. Electron.* **3**, 409–418 (2020).

16. Faria, R., Camsari, K. Y. & Datta, S. Implementing bayesian networks with embedded stochastic MRAM. *AIP Adv.* **8**, 045101 (2018).

17. Sebastian, A., Gallo, M. L. & Eleftheriou, E. Computational phase-change memory: beyond von Neumann computing. *J. Phys. D: Appl. Phys.* **52**, 443002, DOI: 10.1088/1361-6463/ab37b6 (2019).

18. Demirağ, Y. *et al.* PCM-trace: Scalable synaptic eligibility traces with resistivity drift of phase-change materials. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–5 (IEEE, 2021).

19. Payvand, M., Nair, M. V., Müller, L. K. & Indiveri, G. A neuromorphic systems approach to in-memory computing with non-ideal memristive devices: From mitigation to exploitation. *Faraday Discuss.* **213**, 487–510 (2019).

20. Bengel, C. *et al.* Utilizing the switching stochasticity of hfo2/tiox-based reram devices and the concept of multiple device synapses for the classification of overlapping and noisy patterns. *Front. Neurosci.* **15** (2021).

21. Suri, M. *et al.* Bio-inspired stochastic computing using binary CBRAM synapses. *Electron Devices, IEEE Transactions on* **60**, 2402–2409 (2013).

22. Frenkel, C., Legat, J.-D. & Bol, D. MorphIC: A 65-nm 738k-synapse/mm^2 quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning. *IEEE transactions on biomedical circuits systems* **13**, 999–1010 (2019).

23. Kuzum, D., Jeyasingh, R.-G.-D., Lee, B. & Wong, H.-S. P. Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. *Nano Lett.* **12**, 2179–2186, DOI: 10.1021/nl201040y (2012).

24. Covi, E. *et al.* Analog memristive synapse in spiking networks implementing unsupervised learning. *Front. neuroscience* **10**, 1–13 (2016).

25. Ambrogio, S. *et al.* Unsupervised learning by spike timing dependent plasticity in phase change memory (pcm) synapses. *Front. neuroscience* **10** (2016).

26. Payvand, M. & Theogarajan, L. From winner-takes-all to winners-share-all: Exploiting the information capacity in temporal codes. *Neural computation* **30**, 761–791 (2018).

27. Chicca, E. & Indiveri, G. A recipe for creating ideal hybrid memristive-CMOS neuromorphic processing systems. *Appl. Phys. Lett.* **116**, 120501, DOI: 10.1063/1.5142089 (2020).

28. Chicca, E., Stefanini, F., Bartolozzi, C. & Indiveri, G. Neuromorphic electronic circuits for building autonomous cognitive systems. *Proc. IEEE* **102**, 1367–1388 (2014).

29. Vogels, T. P. & Abbott, L. Gating multiple signals through detailed balance of excitation and inhibition in spiking networks. *Nat. neuroscience* **12**, 483 (2009).

30. Machens, C., Wehr, M. & Zador, A. Linearity of cortical receptive fields measured with natural sounds. *The J. neuroscience* **24**, 1089 (2004).

31. Maass, W., Joshi, P. & Sontag, E. Computational aspects of feedback in neural circuits. *PLOS Comput. Biol.* **3**, 1–20 (2007).

32. Dalgaty, T. *et al.* Hybrid CMOS-RRAM neurons with intrinsic plasticity. In *International Symposium on Circuits and Systems (ISCAS), 2019* (IEEE, 2019).

33. Gaba, S., Sheridan, P., Zhou, J., Choi, S. & Lu, W. Stochastic memristive devices for computing and neuromorphic applications. *Nanoscale* **5**, 5872–5878 (2013).

34. Grossi, A. *et al.* Fundamental variability limits of filament-based rram. In *2016 IEEE International Electron Devices Meeting (IEDM)*, 4–7 (IEEE, 2016).

35. Brader, J. M., Senn, W. & Fusi, S. Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Comput.* **19**, 2881–2912 (2007).

36. Dalgaty, T. *et al.* Hybrid neuromorphic circuits exploiting non-conventional properties of RRAM for massively parallel local plasticity mechanisms. *APL Mater.* **7**, 081125, DOI: 10.1063/1.5108663 (2019).

37. Jaeger, H. & Haas, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science* **304**, 78–80 (2004).

38. Maass, W., Natschläger, T. & Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.* **14**, 2531–2560 (2002).

39. Elbrecht, D. & Schuman, C. Neuroevolution of spiking neural networks using compositional pattern producing networks. In *International Conference on Neuromorphic Systems 2020*, 1–5 (2020).

40. Payvand, M., Demirag, Y., Dalgaty, T., Vianello, E. & Indiveri, G. Analog weight updates with compliance current modulation of binary rerams for on-chip learning. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–5 (IEEE, 2020).