The neuromorphic Mosaic: in-memory computing and routing for small-world graphical networks

Thomas Dalgaty[1,†], Filippo Moro[1†], Yiğit Demirağ[2†], Alessio De Pra[1], Giacomo Indiveri[2], Elisa Vianello[1], and Melika Payvand[*2]

[1]CEA, LETI, Université Grenoble Alpes, Grenoble, France
[2]Institute of Neuroinformatics, University of Zurich and ETH Zurich, Zurich, Switzerland
[†]These authors contributed equally to this work.

ABSTRACT

The connectivity in the brain is locally dense and globally sparse - giving rise to a *small-world* graph. This is a principle that has persisted during the evolution of many species - indicating a universal solution to the efficient routing of information. However, existing circuit architectures for artificial neural networks neither leverage this organization nor do they efficiently support small-world neural network models. Here, we propose the *neuromorphic Mosaic*: a non-von Neumann systolic architecture that uses distributed memristors, not only for in-memory computing, but also for in-memory routing, to efficiently implement small-world graph topologies. We design, fabricate, and experimentally demonstrate the building blocks of this architecture, using integrated memristors with 130 nm CMOS technology. We demonstrate that neural networks implemented following this approach can achieve competitive accuracy figures compared to equivalent unconstrained and full-precision networks, for three real-time benchmarks: classification of electrocardiography signals, keyword spotting and motor control via reinforcement learning. The Mosaic shows improvements between one and four orders of magnitude, compared to other event-based neuromorphic architectures for routing events across the network. The Mosaic opens up a new scalable approach for designing edge AI systems based on distributed computing and in-memory routing, offering a natural platform onto which architectures inspired by biological nervous systems can be readily mapped.

Introduction

Despite millions of years of evolution, the fundamental wiring principle of biological brains has been preserved: dense local and sparse global connectivity through synapses between neurons. This persistence indicates the efficiency of this solution in optimizing both computation and the utilization of the underlying neural substrate[1]. Studies have revealed that this connectivity pattern in neuronal networks increases the signal propagation speed[2], enhances echo-state properties[3] and allows for a more synchronized global network[4]. While densely connected neurons in the network are attributed to performing functions such as integration and feature extraction functions[5], long-range sparse connections may play a significant role in the hierarchical organization of such functions[6]. Such neural connectivity is called *small-worldness* in graph theory and is widely observed in the cortical connections of the human brain[2,7,8] (Fig. 1a, b). Small-world connectivity matrices (i.e., a table of connections between neurons) are characterized by a heavily allocated matrix diagonal, with increasingly fewer elements between pairs of neurons the further off-diagonal they are (Fig. 1c).

Crossbar arrays of non-volatile emerging memory technologies e.g., Resistive Random Access Memory (RRAM)[9–14] and Phase Change Memory (PCM)[15,16] have been previously proposed as a means of realizing artificial neural networks on hardware (Fig. 1d). These computing architectures perform in-memory vector-matrix multiplication, the core operation of artificial neural networks - reducing data movement and potentially reducing the power consumption relative to conventional von Neumann architectures[17–22]. However, existing crossbar array architectures are not inherently efficient for realizing small-world neural networks at all scales.

Implementing networks with small-world connectivity in a large crossbar array would result in an under-utilization of the off-diagonal memory elements (i.e., a ratio of non-allocated to allocated connections > 10) - see Fig. 1d and Supplementary Note 1. Furthermore, the impact of analog-related hardware non-idealities such as current sneak-paths, parasitic resistance, and capacitance of the metal lines, as well as excessively large read currents and diminishing yield limit the maximum size of crossbar arrays in practice[23,24]. These issues are also common to biological networks. As the resistance attenuates the spread of the action potential, cytoplasmic resistance sets an upper bound to the length of dendrites[1]. Hence, the intrinsic physical structure of the nervous systems necessitates the use of local over global connectivity.

Drawing inspiration from the biological solution for the same problem leads to a similar optimal silicon layout, a small-world graph. A large crossbar can be divided into an array of smaller, more locally connected crossbars. These correspond to the

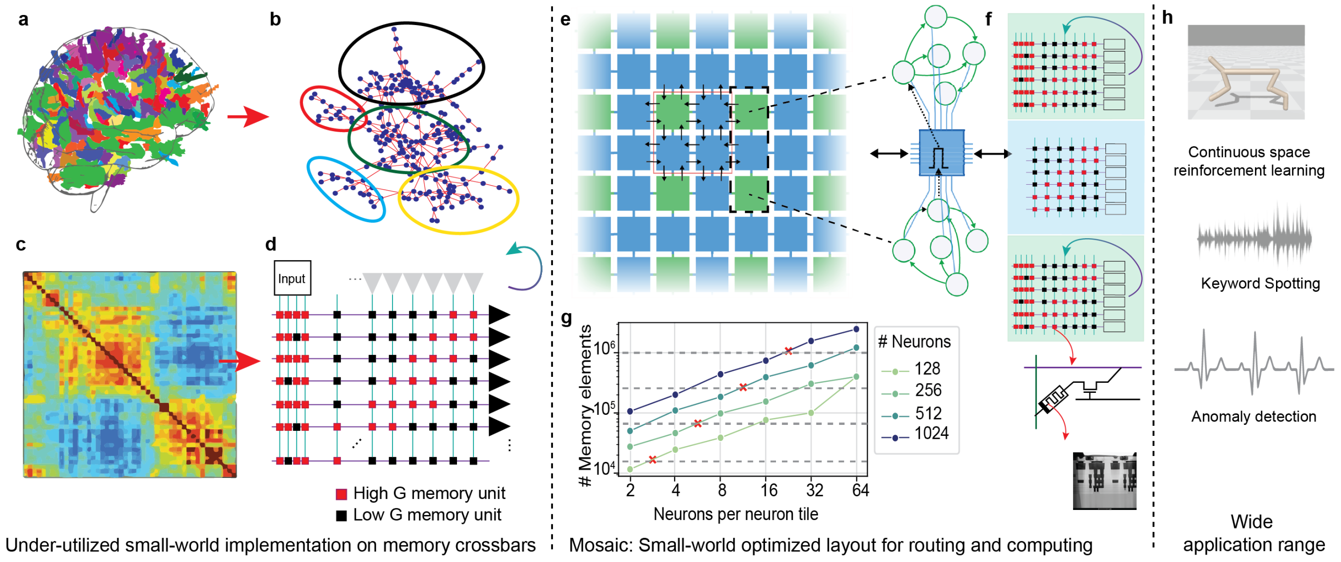

Figure 1. Small-world graphs in biological network and how to build that into a hardware architecture for edge applications. (a) Depiction of small-world property in the brain, with highly-clustered neighboring regions highlighted with the same color. (b) Network connectivity of the brain is a small-world graph, with highly clustered groups of neurons with sparse connectivity among them. (c) (adapted from Bullmore and Sporns 2009[25]). The functional connectivity matrix which is derived from anatomical data with rows and columns representing regions. The diagonal region of the matrix contains the strongest connectivity which represents the connections between the neighboring regions. The off-diagonal elements are not connected. (d) Hardware implementation of the connectivity matrix of b, with neurons and synapses arranged in a crossbar architecture. Black rectangles show the group of memory devices that are never programmed in a small-world network, and are thus "wasted". (e) The Mosaic architecture breaks the large crossbars into small densely-connected crossbars (green *neuron tiles*) and connects them through small routing crossbars (blue *routing tiles*). This gives rise to a distributed two-dimensional mesh, with highly connected clusters of neurons, connected to each other through routers. (f) The state of the resistive memory devices in neuron tiles determines how the information is processed, while the state of the routing devices determines how it is propagated in the mesh. The resistive memory devices are integrated into 130 nm technology. (g) Plot showing the required memory (number of memristors) as a function of the number of neurons per tile, for different total numbers of neurons in the network. The horizontal dashed line indicates the number of required memory bits using a fully-connected RRAM crossbar array. The cross (X) illustrates the cross-over point below which the Mosaic approach becomes favorable. (h) The Mosaic can be used for a variety of edge AI applications, benchmarked here on sensory processing and Reinforcement learning tasks.

green squares of Fig. 1(e). Each green crossbar hosts a cluster of spiking neurons with a high degree of local connectivity. To pass information among these clusters, small routers can be placed between them - the blue tiles in Fig. 1(e). We call this two-dimensional systolic matrix of distributed crossbars, or *tiles*, the neuromorphic *Mosaic* architecture. Each green tile serves as an analog computing core, which sends out information in the form of spikes, while each blue tile serves as a routing core that spreads the spikes throughout the mesh to other green tiles. Thus, the Mosaic takes advantage of distributed and de-centralized computing and routing to enable not only in-memory computing, but also *in-memory routing* (Fig. 1(f)).

Neighborhood-based computing with resistive memory has been previously explored through using Cellular Neural Networks[26,27], Self-organizing Maps (SOM)[28], and the cross-net architecture[29]. Though cellular architectures use local clustering, their lack of global connectivity limits both the speed of information propagation and their configurability. Therefore their application has been mostly limited to low-level image processing[30]. This is true also for SOMs, which exploit neighboring connectivity and are typically trained with unsupervised methods to visualize low-dimensional data[31]. Similarly, the crossnet architecture proposed to use distributed small tilted integrated crossbars on top of the Complementary Metal-Oxide-Semiconductor (CMOS) substrate, to create local connectivity domains for image processing[29]. The tilted crossbars allow the nano-wire feature size to be independent of the CMOS technology node[32], however this approach requires extra post-processing lithographic steps in the fabrication process.

Unlike most previous approaches, the Mosaic supports both dense local connectivity, and globally sparse long-range connections, by introducing re-configurable routing crossbars between the computing tiles. This allows to flexibly program specific small-world network configurations and to compile them onto the Mosaic for solving the desired task. Moreover, the

Mosaic is fully compatible with standard integrated RRAM/CMOS processes available at the foundries, without the need for extra post-processing steps. The Mosaic also goes beyond the state of the art in the routing scheme of standard CMOS Spiking Neural Network (SNN) hardware based on the Address-Event Representation (AER)[33–35], by removing the need to store each neuron's connectivity information in either bulky local or centralized memory units which draw static power and can consume a large chip area (Supplementary Note 2).

In this Article, we first present the Mosaic architecture. We report electrical circuit measurements from computational and routing tiles that we designed and fabricated in 130 nm CMOS technology co-integrated with Hafnium dioxide-based RRAM devices. Then, calibrated on these measurements, and using a novel method for training small-world neural networks that exploits the intrinsic layout of the Mosaic, we run system-level simulations applied to a variety of edge computing tasks (Fig. 1h). Finally, we compare our approach to other neuromorphic hardware platforms which highlights an impressive reduction in event routing energy - between one and four orders of magnitude.

Results

In the Mosaic (Fig. 1d), each of the tiles consists of a small memristor crossbar that can receive and transmit spikes to and from their neighboring tiles to the North (N), South (S), East (E) and West (W) directions (Supplementary Note 3). The memristive crossbar array in the green neuron tiles stores the synaptic weights of several Leaky Integrate and Fire (LIF) neurons. These neurons are implemented using analog circuits and are located at the termination of each row, emitting voltage spikes at their outputs[36]. These spikes are communicated between neuron tiles through a mesh of blue routing tiles, whose crossbar array stores the connectivity pattern between neuron tiles. Together, the two tiles give rise to a continuous *mosaic* of neuromorphic computation and memory for realizing spiking small-world neural networks.

Small-world neural network topology can be obtained by randomly programming memristors in a computer model of the Mosaic (see Methods and Supplementary Note 4). The resulting graph exhibits an intriguing set of connection patterns that reflect those found in many of the small-world graphical motifs observed in animal nervous systems. For example, central 'hub-like' neurons with connections to numerous nodes, reciprocal connections between pairs of nodes reminiscent of winner-take-all mechanisms, and some heavily connected local neural clusters[8]. If desired, these graph properties could be adapted on the fly by re-programming the RRAM states in the two tile types. For example, a set of desired small-world graph properties can be achieved by randomly programming the RRAM devices into their High-Conductive State (HCS) with a certain probability (Supplementary Note 4). Random programming can for example be achieved elegantly by simply modulating RRAM programming voltages[37].

For Mosaic-based small-world graphs, we estimate the required number of memory devices (synaptic weight and routing weight) as a function of the total number of neurons in a network, through a mathematical derivation (see Methods). Fig. 1(g) plots the memory footprint as a function of the number of neurons in each tile for different network sizes. Horizontal dashed lines show the number of memory elements using one large crossbar for each network size, as has previously been used for Recurrent Neural Networks (RNN) implementations[38]. The cross-over points, at which the Mosaic memory footprint becomes favorable, are denoted with a cross. While for smaller network sizes (here 128 neurons) no memory reduction is observed compared to a single large array, the memory saving becomes increasingly important as the network is scaled. For example, given a network of 1024 neurons and 4 neurons per neuron tile, the Mosaic requires almost one order of magnitude fewer memory devices than a single crossbar to implement an equivalent network model.

Neuron tile circuits: small-worlds

Each neuron tile in the Mosaic (Fig. 2a) is composed of multiple rows, a circuit that models a LIF neuron and its synapses. The details of one neuron row is shown in Fig. 2b. It has N parallel one-transistor-one-resistor (1T1R) RRAM structures at its input. The synaptic weights of each neuron are stored in the conductance level of the RRAM devices in one column. On the arrival of any of the input events $V_{in<i>}$, the amplifier pins node V_x to V_{top} and thus a read voltage equivalent to $V_{top} - V_{bot}$ is applied across G_i, giving rise to current i_{in} at M_1, and in turn to i_{buff}. This current pulse is fed to the "synaptic dynamics" circuit, Differential Pair Integrator (DPI)[39] which low pass filters it through charging the capacitor M_6 in the presence of the pulse, and discharging it through current I_{tau} in the absence of the pulse, thus generating an exponentially decaying current. The current dynamics, I_{syn}, is injected into the neuron's membrane potential node, V_{mem}, and charges capacitor M_{13}, leaking through $M_1 1$, whose rate is controlled by V_{lk} at its gate. As soon as the voltage developed on V_{mem} passes the threshold of the following inverter stage, it generates a pulse. The refractory period time constant depends on the MOS cap $M_1 6$ and the bias on V_{rp}. (For a detailed explanation of the circuit, please see Supplementary Note 5). Following a systolic organization[40], each input or output spike can enter from, and exit towards, the neighboring N, S, E, W tiles (Supplementary Note 6), as shown in the inset of Fig. 2a.

We have fabricated and measured the circuits of the neuron tile in a 130 nm CMOS technology integrated with RRAM devices[41]. In the fabricated circuit, we statistically characterized the RRAMs, resulting in eight stable conductance states shown

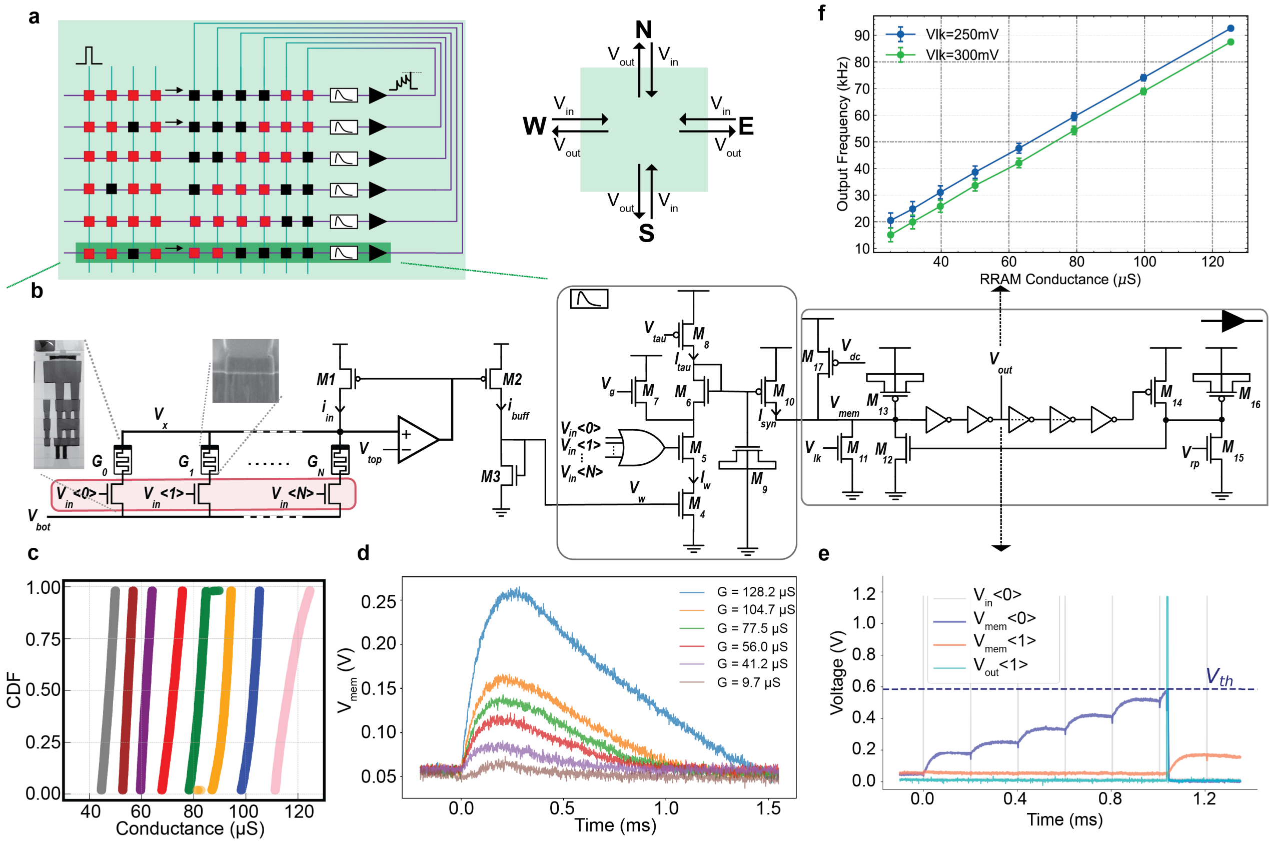

Figure 2. Experimental results from the neuron column circuit. (a) The neuron tile, a crossbar with feed-forward and recurrent inputs. (b) Detailed schematic of one row of the fabricated crossbars. RRAMs are used as the weights of the neurons. Insets of (left) scanning electron and (right) transmission electron microscopy images respectively show cross-sections of the 1T1R stack and the hafnium-dioxide layer sandwiched between top and bottom memristor electrodes. On the arrival of any of the input events $V_{in<i>}$, read voltage equivalent to $V_{top} - V_{bot}$ is applied across G_i, giving rise to a current pulse i_{in}, and in turn to i_{buff}, feeding to the synaptic dynamic block. The "Differential Pair Integrator" low-pass filters the current pulse, giving rise to an exponentially decaying current dynamic, whose time constant is a function of the MOS capacitor M_9 and I_{tau}. The current is then fed and integrated by the neuron's membrane potential, V_{mem}, and upon passing the threshold of the inverter creates an output pulse (V_{out}). The refractory period time is determined by the MOS cap M_{16} and the bias on V_{rp}. (c) Eight cumulative distributions resulting from the application of a single SET programming pulse on each device in an array of 4096 RRAM devices over a range of SET programming currents, I_{SET}. (d) From an initial resting membrane voltage of 0.05V, the membrane voltage waveform recorded by an oscilloscope is plotted in time due to the arrival of a single input pulse. The conductance of the device being read is swept from 10μS to 125μS and the resulting waveforms are measured for each conductance value. (e) Due to an input pulse train (gray pulses) at $V_{in} < 0 >$ the V_{mem} of the zeroth neuron integrates an increasing amount of voltage (purple trace) until, after six pulses, the neuron fires (light blue trace). As a result of the feedback connection to the other neuron column, neuron 1 also exhibits an increase in its membrane voltage. (f) The output frequency of the neuron is a linear function of the conductance of the RRAM. The error bar reflects the variability across 4096 devices.

in Fig. 2c. After programming each device of Fig. 2b, we applied an input pulse to $V_{in} < 0 >$ and measured the signal V_{mem}. This is plotted in Fig. 2d, illustrating that the increase in RRAM conductance increases the peak V_{mem} voltage resulting from a single input pulse. Thus, the RRAM serves well as a programmable synaptic weight element. If multiple pulses arrive to $V_{in} < 0 >$, V_{mem} increases until it reaches the threshold of 0.6 V (Fig. 2e). We have characterized the firing rate of the neuron as a function of the conductance of the RRAM in Fig. 2f, which shows a linear behavior, as expected. The error bars on the plot are due to the variability of the devices in the 4 kb array.

Routing tile circuits: connecting small-worlds

A routing tile circuit is shown in Fig. 3a. It acts as a flexible means of configuring how spikes emitted from neuron tiles propagate locally between small-worlds. The functional principles of the routing tile circuits are similar to the neuron tiles. The principal difference is the replacement of the biological synapse and neuron circuit models with a simple current comparator

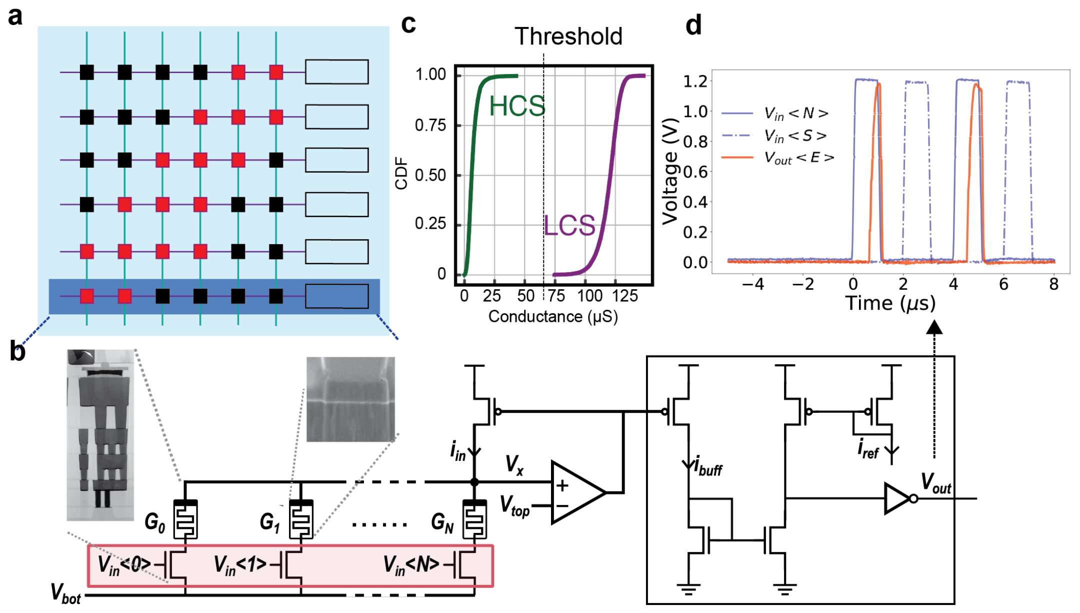

Figure 3. Experimental measurements of the fabricated routing tile circuits. (a) The routing tile, a crossbar whose memory state steers the input spikes towards the destination. (b) Detailed schematic of one row of the fabricated routing circuits. On the arrival of a spike to any of the input ports of the routing tile, $V_{in} < i >$, a current proportional to G_i flows in i_{in}, similar to the neuron tile. This current is compared against a reference current, with a current comparator. The spike will get regenerated, thus "pass", if $i_{in} > i_{ref}$, or is "blocked" otherwise. (c) Measurements from 4 kb array shows the Cumulative Distribution Function (CDF) of the RRAM in its High Conductive State (HCS) and Low Conductive State (LCS). The line between the distributions that separates the two is considered as the "Threshold", giving rise to I_{ref}, which determines if the input spikes are passed or blocked. (d) Experimental results from the routing tile, with continuous and dashed blue traces showing the waveforms applied to the N and S inputs while the orange trace shows the response of the output towards the E port. The E output port follows the N input resulting from the device programmed into the HCS, while the input from the S port gets blocked as the corresponding RRAM device is in its LCS.

circuit (highlighted with a box in Fig. 3b). On the arrival of a spike on an input port of the routing tile, $V_{in} < i >, 0 < i < N$, a current proportional to G_i flows to the device, giving rise to read current i_{buff}. A current comparator, highlighted in the box, compares i_{buff} against i_{ref}, and regenerates a spike if it is higher, and blocks it if it is lower, and the output remains at zero. Therefore, the state of the device serves to either pass or block input spikes arriving from different input ports (N,S,W,E), sending them to its output ports (N,S,W,E).

Using a fabricated routing tile circuit, we demonstrate its functionality experimentally in Fig. 3d. Continuous and dashed blue traces show the waveforms applied to the N and S inputs of the tile respectively, while the orange trace shows the response of the output towards the E port. The E output port follows the N input resulting from the corresponding RRAM programmed into itsHCS, while the input from the S port gets blocked as the corresponding RRAM device is in its LCS, and thus the output remains at zero. This output pulse propagates onward to the next tile. Note that in Fig. **??** the output spike does not appear as rectangular due to the large capacitive load of the probe station (see Methods). To allow for greater reconfigurability, more channels per direction can be used in the routing tiles (see Supplementary Fig. S4).

Application to real-time sensory-motor processing

The neuromorphic Mosaic is a programmable hardware well suited for the application of pre-trained small-world Recurrent Spiking Neural Network (RSNN) in energy and memory-constrained applications at the edge. We assess the suitability of the Mosaic on a series of representative tasks, including anomaly detection in heartbeat (application in wearable devices), keyword spotting (application in voice command), and motor system control (application in robotics) tasks (Fig. 4 a,b,c respectively). To do so, we apply these tasks to three network cases, (i) a non-constrained RSNN with full-bit precision weights (32 bit Floating Point (FP32)) (Fig. 4 d), (ii) Mosaic constrained connectivity with FP32 weights (Fig. 4 e), and (iii) Mosaic constrained connectivity with noisy and quantized RRAM weights (Fig. 4 f).

For training case i, we use Backpropagation Through Time (BPTT)[42] with surrogate gradient approximations of the derivative of a LIF neuron activation function on a vanilla RSNN[43] (see Methods). For training case (ii), we introduce a Mosaic-regularized cost function during the training, which leads to a learned weight matrix with small-world connectivity (see Methods for details). For case (iii), we quantize the weights using a mixed hardware-software experimental methodology whereby memory elements in a Mosaic software model are assigned conductance values programmed into a corresponding memristor in a fabricated array. Programmed conductances are obtained through a closed-loop programming strategy[44–47].

For all the networks and tasks, the input is fed as a spike train and the output class is identified as the neuron with the highest firing rate. The RSNN of case (i) includes a standard input layer, recurrent layer, and output layer. In the Mosaic cases (ii) and (iii), the inputs are directly fed into the Mosaic neuron tiles from the top left, are processed in the small-world RSNN, and the resulting output is taken directly from the opposing side of the Mosaic, by assigning some of the neuron tiles in the Mosaic as output neurons. Relative to the RSNN, this scheme avoids the need for input and output readout layers that may greatly simplify a practical implementation.

Electrocardiography (ECG) anomaly detection We first benchmark our approach on a binary classification task in detecting anomalies in the ECG recordings of the MIT-BIH Arrhythmia Database[49]. In order for the data to be compatible with the RSNN, we first encode the continuous ECG time-series into trains of spikes using a delta-modulation technique, which describes the relative changes in signal magnitude[50,51] (see Methods). An example heartbeat and its spike encoding are plotted in Fig. 4a.

The accuracy over the test set for five iterations of training, transfer, and test for cases (i) (red), (ii) (green) and (iii) (blue) is plotted in Fig. 4g using a boxplot. Although the Mosaic constrains connectivity to follow a small-world pattern, the median accuracy of case (ii) only drops by 3% compared to the non-constrained RSNN of case (i). Introducing the quantization and noise of the RRAM devices in case (iii), drops the median accuracy further by another 2%, resulting in a median accuracy of 92.4%. As often reported, the variation in the accuracy of case iii also increases due to the cycle-to-cycle variability of RRAM devices.

Keyword spotting (KWS) We then benchmarked our approach on a 20-way speech classification task using the Spiking Heidelberg Dataset (SHD)[48] dataset. SHD includes the spoken digits between zero and nine in English and German uttered by 12 speakers. In this dataset, the speech signals have been encoded into spikes using a biologically-inspired cochlea model which effectively computes a spectrogram with Mel-spaced filter banks, and convert them into instantaneous firing rates[48].

The accuracy over the test set for five iterations of training, transfer, and test for cases (i) (red), (ii) (green) and (iii) (blue) is plotted in Fig. 4h using a boxplot. The dashed red box is taken directly from the SHD paper[48]. Again, the Mosaic connectivity constraints have only an effect of about 2.5% drop in accuracy, and a further drop of 1% when introducing RRAM quantization and noise constraints.

Motor control by Reinforcement Learning Finally, we also benchmark the Mosaic in a motor system control Reinforcement Learning (RL) task, i.e., the *Half-cheetah*[52]. RL has applications ranging from active sensing via camera control[53] to dexterous robot locomotion[54].

To train the network weights, we employ the evolutionary strategies (ES) from Salimans et. al.[55] in reinforcement learning settings[56–58]. ES enables stochastic perturbation of the network parameters, evaluation of the population fitness on the task and updating the parameters using stochastic gradient estimate, in a scalable way for RL.

Fig. 4i shows the maximum gained reward for five runs in cases i, ii, and iii, which indicates that the network learns an effective policy for forward running. Unlike tasks a and b, the network connectivity constraints and parameter quantization have relatively little impact.

Encouragingly, across three highly distinct tasks, performance was only slightly impacted when passing from a unconstrained neural network topology to a small-world neural network. In particular, for the half-cheetah RL task, this had no impact.

Neuromorphic platform routing energy

In-memory computing allows for the energy consumption inherent to data movement in Von-Neumann architectures to be greatly reduced. Although crossbars bring memory and computing together, when neural networks are scaled up, neuromorphic hardware will require an array of distributed crossbars (or cores) when faced with physical constraints, such as IR drop and capacitive charging. Small-world networks may naturally permit the minimization of communication between these crossbars, but a certain energy and latency cost associated with the data movement will remain, since the compilation of the small-world network on a general-purpose architecture is not ideal. Hardware specifically designed for small-world networks will ideally minimize these energy and latency costs 1 g. In order to understand how the routing efficiency of the Mosaic compares to other neuromorphic hardware platforms, with reported energy values, we (i) compare the energy and latency of routing one event within a core (0-hop), (ii) to the neighboring core (1-hop) and (iii) the total routing power consumption required in task A (Fig. 4a). The results are presented in Table 1 where we have scaled the energy and latency figures of all the platforms to 130 nm technology using general scaling laws[63] for a fair comparison. The scaled energy figures show that although the Mosaic's design has not been optimized for energy efficiency, the 0- and 1-hop routing energy is significantly reduced relative to other approaches by between one and four orders of magnitude. This efficiency can be attributed to the Mosaic's *in-memory routing* approach resulting in low-energy routing memory access which is distributed in space. This reduces the size of each

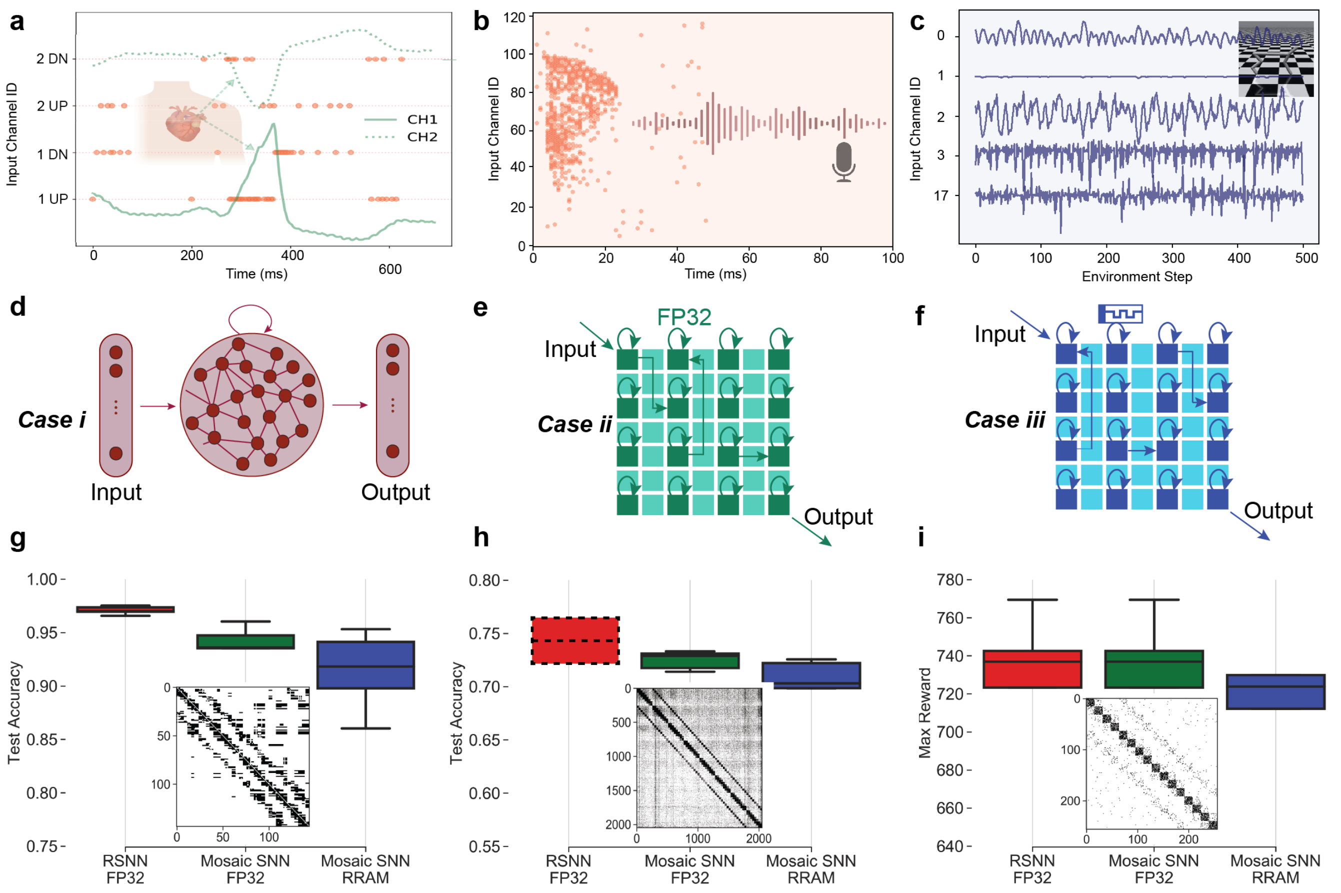

Figure 4. Benchmarking the Mosaic against three edge tasks, heartbeat arrhythmia detection, keyword spotting, and motor control by reinforcement learning. (a,b,c) A depiction of the three tasks, along with the corresponding input presented to the Mosaic. (a) shows the ECG task, where each of the two-channel waveforms is encoded into up (UP) and down (DN) spiking channels, representing the signal derivative direction. (b) shows the KWS tasks with the spikes representing the density of information in different input (frequency) channels. (c) shows the half-cheetah RL task with input channels representing state space, consisting of positional values of different body parts of the cheetah, followed by the velocities of those individual parts.) (d,e,f) A representation of the three network cases applied to each task. (d) Case (i) is a non-constrained Recurrent Spiking Neural Network (RSNN) with full-bit precision weights (FP32). RSNN includes an input layer, a recurrent layer and an output layer. (e) Case (ii) is a Mosaic-constrained connectivity with FP32 weights, without explicit input and output layers. The input is fed directly into the Mosaic and the output is taken directly from the Mosaic. The circular arrows indicate the local recurrent connections, and the straight arrows are sparse global connections between the cores. (f) Case (iii) is similar to case (ii), only with noisy and quantized RRAM weights. The resulting connectivity matrix after training on each task clearly shows the small-world graph property implementable by the Mosaic. (g,h,i) A comparison of the accuracy of the three tasks between the case i (red), case ii (green), and case iii (blue). Box-plots show the accuracy/ maximum reward over five iterations and the colored boxes span the upper and lower quartiles of accuracy, while the upper and lower whiskers extend to the maximum and minimum accuracies/maximum reward obtained over the five iterations. The median accuracy is indicated with a solid horizontal line. The dashed red box for the KWS task with FP32 RSNN network is the results taken from Cramer et al, 2020[48] with 512 neurons for comparison. This comparison shows that the drop in accuracy due to the Mosaic connectivity and further due to RRAM weights is not considerable for any of the tasks. The inset figures show the resulting Mosaic connectivity after training, which follows a small-world graphical connectivity.

router, compared to larger centralized routers in other platforms, and thus reduces the access energy. Moreover, it avoids the use of Content Addressable Memorys (CAMs), used in many spike routing platforms, which consumes the majority of routing energy in other neuromorphic platforms (Supplementary Note 2). The Mosaic's latency figure per router is comparable to the average latency of other platforms. Often, and in particular, for neural networks with sparse firing activity, this is a negligible factor. In applications requiring sensory processing of real-world slow-changing signals, the time constants dictating how quickly the model state evolves will be determined by the size of the resistor-capacitor circuit in Fig. **??**- typically on the order

Table 1. Comparison of routing performance

Neuromorphic Chip	TrueNorth[59]	SpiNNaker[60]	Neurogrid[61]	Dynap-SE[34]	Loihi[62]	Mosaic
Techn.	28 nm	130 nm	180 nm	180 nm	14 nm	130 nm
Routing	on-chip	on-chip	on/off-chip	on-chip	on-chip	on-chip
0-hop Energy normalized to 130 nm	6 pJ	30.0 nJ	160 pJ	13.4 pJ	60.4 pJ	**400 fJ** $^\circ$
1-hop Routing Energy Scaled to 130 nm	5.52 pJ	4 nJ	8.35 nJ	160 pJ	10.24 pJ	**1.6 pJ** $^\circ$
1-hop Routing Latency Scaled to 130 nm	29 ns	**200 ps**	14.4 ns	28.88 ns	60.35 ns	25 ns
Optimized for Small-world connectivity	No	No	No	Yes	No	Yes
Routing power task A	3.27 nW	5.06 μW	4.30 μW	78.1 nW	11.2 nW	**809pW** $^\circ$

*

$^\circ$ Assuming an average resistance value of 10 kΩ, and a read pulse 10 ns width.

of tens or hundreds of milliseconds.

The final row of Table 1 indicates the power consumption of all of the hardware in task A (ECG anomaly detection). All hardware is assumed to use a core (i.e., neuron tile) size of 32 neurons, and to have an N-hop energy cost equal to N times the 1-hop value. The potential of the Mosaic is clearly demonstrated, whereby a power consumption of only a few hundreds of pico Watts is required, relative to a few nano/microwatts in the other neuromorphic platforms.

Conclusion

We have proposed the neuromorphic Mosaic, a novel neuromorphic computing architecture based on a systolic array of small memristor crossbars. Crucially, the Mosaic uses distributed non-volatile resistive memory devices in an analog fashion, not only for computation, but also for in-memory spike routing between neural compute elements. Electrical circuit measurements of fabricated circuit blocks were reported that offer a first validation of the approach. We empirically quantified the impact of both the small-world neural network topology and low memristor precision on three diverse and challenging tasks representative of edge AI settings. We also introduced an adapted machine learning strategy which enforces small-worldness and accounted for this low-precision. The results achieved across these tasks were comparable to those achieved by floating point precision models with an unconstrained network connectivity. Most striking was perhaps the massive reduction in routing power consumption of the neuromorphic Mosaic relative to state of the art neuromorphic computing platforms. Graph-based computing is currently receiving attention as a promising means of leveraging the capabilities of SNNs[64]. The Mosaic is thus a timely dedicated hardware architecture optimized for a specific type of graph that is abundant in nature and in the real-world and that promises to find application at the extreme-edge.

Methods

Design, fabrication of Mosaic circuits

Neuron and routing column circuits

Both neuron and routing column share the common front-end circuit in Fig. **??**(highlighted in gray) which reads the conductances of the RRAM devices. The RRAM bottom electrode has a constant DC voltage V_{bot} applied to it and the common top electrode is pinned to the voltage V_x by a rail-to-rail operational amplifier (OPAMP) circuit. The OPAMP output is connected in negative feedback to its non-inverting input (due to the 90 degrees phase-shift between the gate and drain of transistor M_1 in Fig. **??**) and has the constant DC bias voltage V_{top} applied to its inverting input. As a result, the output of the OPAMP will modulate the gate voltage of transistor M_1 such that the current it sources onto the node V_x will maintain its voltage as close as possible to the DC bias V_{top}. Whenever an input pulse $V_{in} < n >$ arrives, a current i_{in} equal to $(V_x - V_{bot})G_n$ will flow out of the bottom electrode.

The negative feedback of the OPAMP will then act to ensure that $V_x = V_{top}$, by sourcing an equal current from transistor M_1. By connecting the OPAMP output to the gate of transistor M_2, a current equal to i_{in}, will therefore also be buffered, as i_{buff}, into the branch composed of transistors M_2 and M_3 in series. In the routing tile, this current is compared against a reference current, and if higher, a pulse is generated and transferred onwards. The current comparator circuit is composed of two current mirrors and an inverter (see Supplementary Fig. S8). In the neuron column, this current is injected into a CMOS differential-pair integrator synapse circuit model[65] which generates an exponentially decaying waveform from the onset of the pulse with an amplitude proportional to the injected current. Finally, this exponential current is injected onto the membrane capacitor of a CMOS leaky-integrate and fire neuron circuit model[66] where it integrates as a voltage (see Supplementary Fig. S4). Upon exceeding a voltage threshold (the switching voltage of an inverter) a pulse is emitted at the output of the circuit. This pulse in turn feeds back and shunts the capacitor to ground such that it is discharged. Further circuits were required in order to program the device conductance states. Notably, multiplexers were integrated on each end of the column in order to be able to apply voltages to the top and bottom electrodes the RRAM devices.

Fabrication/integration

The circuits described in the Results section have been taped-out in 130 nm technology at CEA-Leti, in a 200 mm production line. The Front End of the Line, below metal layer 4, has been realized by ST-Microelectronics, while from the fifth metal layer 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 a $TiN/HfO_2/Ti/TiN$ stack. Each device is accessed by a transistor giving rise to the 1T1R unit cell. The size of the access transistor is $650\,nm$ wide. 1T1R cells are integrated with CMOS-based circuits by stacking the RRAM cells on the higher metal layers. In the cases of the neuron and routing tiles, 1T1R cells are organized in a small - either 2x2 or 2x4 - matrix in which the bottom electrodes are shared between devices in the same column and the gates shared with devices in the same row. In this way, the devices can be accessed in a parallel manner. The circuits integrated into the wafer, were accessed by a probe card which connected to the pads of the dimension of $[50x90]\mu m^2$.

RRAM characteristics

Resistive switching in the devices used in our paper are based on the formation and rupture of a filament as a result of the presence of an electric field that is applied across the device. The change in the geometry of the filament results in different resistive state in the device. 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 HCS, otherwise the cell is is the Low-Conductive State (LCS). For a SET operation, the bottom of the 1T1R structure is conventionally left at ground level, and a positive voltage is applied to the 1T1R top electrode. 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 $4.5V$.

Mosaic circuit measurement setups

The tests involved analyzing and recording the dynamical behavior of analog CMOS circuits as well as programming and reading RRAM devices. Both phases required dedicated instrumentation, all simultaneously connected to the probe card. For programming and reading the RRAM devices, Source Measure Units (SMU)s from a Keithley 4200 SCS machine were used. To maximize stability and precision of the programming operation, SET and RESET are performed in a quasi-static manner. This means that a slow rising and falling voltage input is applied to either the Top (SET) or Bottom (RESET) electrode, while the gate is kept at a fixed value. To the $V_{top}(t)$, $V_{bot}(t)$ voltages, we applied a triangular pulse with rising and falling times of 1 sec and picked a value for V_{gate}. For a SET operation, the bottom of the 1T1R structure is conventionally left at ground level, while in the RESET case the V_{top} is equal to 0 V and a positive voltage is applied to V_{bot}. Typical values for the SET operation are V_{gate} in $[0.9-1.3]\,V$, while the V_{top} peak voltage is normally at $2.0\,V$. Such values allow to modulate the RRAM resistance in an interval of $[5-30]\,k\Omega$ corresponding to the HCS of the device. 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 LCS is less controllable than the HCS due to the inherent stochasticity related to the rupture of the conductive filament, thus the HRS level is spread out in a wider $[80-1000]\,k\Omega$ interval. 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 $4.5\,V$.

Inputs and outputs are analog dynamical signals. In the case of the input, we have alternated two HP 8110 pulse generators with a Tektronix AFG3011 waveform generator. As a general rule, input pulses had a pulse width of $1\,\mu s$ and rise/fall time of $50\,ns$. This type of pulse is assumed as the stereotypical spiking event of a Spiking Neural Network. Concerning the outputs, a $1\,GHz$ Teledyne LeCroy oscilloscope was utilized to record the output signals.

Mosaic layout aware training via regularizing the loss function

We introduce a new regularization function, L_M, that emphasizes the realization cost of short and long-range connections in the Mosaic layout. Assuming the neuron tiles are placed in a square layout, L_M calculates a matrix $H \in \mathbb{R}^{j \times i}$, expressing the minimum number of routing tiles used to connect a source neuron N_j to target neuron N_i, based on their neuron tile positions on Mosaic. Following this, a static mask $S \in \mathbb{R}^{j \times i}$ is created to exponentially penalize the long-range connections such that $S = e^{\beta H} - 1$, where β is a positive number that controls the degree of penalization for connection distance. Finally, we calculate the $L_M = \sum S \odot W^2$, for the recurrent weight matrix $W \in \mathbb{R}^{j \times i}$. Note that the weights corresponding to intra-neuron tile connections (where $H = 0$) are not penalized, allowing the neurons within a neuron tile to be densely connected. During the training, task-related cross-entropy loss term (total reward in case of RL) increases the network performance, while L_M term reduces the strength of the neural network weights creating long-range connections in Mosaic layout. Starting from the 10th epoch, we deterministically prune connections (replacing the value of corresponding weight matrix elements to 0) when their L_1-norm is smaller than a fixed threshold value of 0.005. This pruning procedure privileges local connections (i.e., those within a neuron tile or to a nearby neuron tile) and naturally gives rise to a small-world neural network topology. Our experiments found that gradient norm clipping during the training and reducing the learning rate by a factor of ten after 135th epoch in classification tasks help stabilize the optimization against the detrimental effects of pruning.

RRAM-aware noise-resilient training

Offline training of neural networks results in full-precision weights that hinder the performance when deployed on RRAM crossbar arrays due to analog-related non-idealities such as programming stochasticity, temporal conductance relaxation, and read noise[44–47]. To mitigate this detrimental effects at the weight transfer stage, we adapted the noise-resilient training method for RRAM devices[67,68]. Similar to quantization-aware-training, at every forward pass, the original network weights are altered via additive noise (quantized) using a straight through estimator. We used a Gaussian noise with zero mean and standard deviation equal to 5% of the maximum conductance to emulate transfer non-idealities. The profile of this additive noise is based on our RRAM characterization of an array of 4096 RRAM devices[44], which are programmed with a program-and-verify scheme (up to 10 iterations) to various conductance levels then measured after 60 seconds for modeling the resulting distribution.

ECG task description

The Mosaic hardware-aware training procedure is tested on a electrocardiogram arrhythmia detection task. The ECG dataset was downloaded from the MIT-BIH arrhythmia repository[49]. The database is composed of continuous 30-minute recordings measured from multiple subjects. The QRS complex of each heartbeat has been annotated as either healthy or exhibiting one of many possible heart arrhythmias by a team of cardiologists. We selected one patient exhibiting approximately half healthy and half arrhythmic heartbeats. Each heartbeat was isolated from the others in a 700 ms time-series centered on the labelled QRS complex. Each of the two 700 ms channel signals were then converted to spikes using a delta modulation scheme[69]. This consists of recording the initial value of the time-series and, going forward in time, recording the time-stamp when this signal changes by a pre-determined positive or negative amount. The value of the signal at this time-stamp is then recorded and used in the next comparison forward in time. This process is then repeated. For each of the two channels this results in four respective event streams - denoting upwards and downwards changes in the signals. During the simulation of the neural network, these four event streams corresponded to the four input neurons to the spiking recurrent neural network implemented by the Mosaic.

Data points were presented to the model in mini-batches of 16. Two populations of neurons in two neuron tiles were used to denote whether the presented ECG signals corresponded to a healthy or an arrhythmic heartbeat. The softmax of the total number of spikes generated by the neurons in each population was used to obtain a classification probability. The negative log-likelihood was then minimized using the categorical cross-entropy with the labels of the signals.

Keyword Spotting task description

For keyword spotting task, we used SHD dataset (20 classes, 8156 training, 2264 test samples). Each input example drawn from the dataset is sampled three times along the channel dimension without overlap to obtain three augmentations of the same data with 256 channels each. The advantage of this method is that it allows feeding the input stream to fewer neuron tiles by reducing the input dimension and also triples the sizes of both training and testing datasets. We set the simulation time step to 1 ms in our simulations. The recurrent neural network architecture consists of 2048 LIF neurons with 45 ms membrane time constant. The neurons are distributed into 8x8 neuron tiles with 32 neurons each. The input spikes are fed only into the neurons of the Mosaic layout's first row (8 tiles). The network prediction is determined after presenting each speech data for 100 ms by counting the total number of spikes from 20 neurons (total number of classes) in 2 output neuron tiles located in the bottom-right of the Mosaic layout. The neurons inside input and output neuron tiles are not recurrently connected. The network is trained using BPTT on the loss $L = L_{CE} + \lambda L_M$, where L_{CE} is the cross-entropy loss between input logits and target and L_M is the Mosaic-layout aware regularization term. We use batch size of 512 and suitably tuned hyperparameters.

Reinforcement Learning task description

In the RL experiments, we test the versatility of a Mosaic-optimized RSNN on a continuous action space motor-control task, *half-cheetah*, implemented using the BRAX physics engine for rigid body simulations[70]. At every timestep t, environment provides an input observation vector $o^t \in \mathbb{R}^{25}$ and a scalar reward value r^t. The goal of the agent is to maximize the expected sum of total rewards $R = \sum_{t=0}^{1000} r^t$ over an episode of 1000 environment interactions by selecting action $a^t \in \mathbb{R}^7$ calculated by the output of the policy network. The policy network of our agent consists of 256 recurrently connected LIF neurons, with a membrane decay time constant of 30 ms. The neuron placement is equally distributed into 16 neuron tiles to form a 7x7 Mosaic layout. At each time step, the observation vector o^t is accumulated into the membrane voltages of the first 25 neurons of two upper left input tiles. Furthermore, action vector a^t is calculated by reading the membrane voltages of the last seven neurons in the bottom right corner after passing *tanh* non-linearity.

We considered Evolutionary Strategies (ES) as an optimization method to adjust the RSNN weights such that after the training, the agent can successfully solve the environment with a policy network with only locally dense and globally sparse connectivity. We found ES particularly promising approach for hardware-aware training as (i) it is blind to non-differentiable hardware constraints e.g., spiking function, quantizated weights, connectivity patterns, and (ii) highly parallelizable since ES does not require spiking variable to be stored for thousand time steps compared to BPTT that explicitly calculates the gradient. In ES, the fitness function of an offspring is defined as the combination of total reward over an episode, R and realization cost of short and long-range connections L_M (same as KWS task), such that $F = R - \lambda L_M$. We used the population size of 4096 (with antithetic sampling to reduce variance) and mutation noise standard deviation of 0.05. At the end of each generation, the network weights with L_0-norm smaller than a fixed threshold are deterministically pruned. The agent is trained for 1000 generations.

Calculation of memory footprint

We calculate the Mosaic architecture's Memory Footprint (MF) in comparison to a large crossbar array, in building small-world graphical models.

To evaluate the MF for one large crossbar array, the total number of devices required to implement any possible connections between neurons can be counted - allowing for any Spiking Recurrent Neural Networks (SRNN) to be mapped onto the system. Setting N to be the number of neurons in the system, the total possible number of connections in the graph is $MF_{ref} = N^2$.

For the Mosaic architecture, the number of RRAM cells (i.e., the MF) is equal to the number of devices in all the neuron tiles and routing tiles: $MF_{mosaic} = MF_{NeuronTiles} + MF_{RoutingTiles}$.

Considering each neuron tile with k neurons, each neuron tile contributes to $4 \times k^2$ devices (where the factor of 4 accounts for the four possible directions to which each tile can connect). Evenly dividing the N total number of neurons in each neuron tile gives rise to $T = ceil(N/k)$ required neuron tiles. This brings the total number of devices attributed to the neuron tile to $T \times 4 \times k^2$.

The number of routing tiles which connects all the neuron tiles depends on the geometry of the Mosaic systolic array. Here, we assume neuron tiles assembled in a square, each with a routing tile on each side. We consider R to be the number of routing tiles with $4k^2$ devices in each. This brings the total number of devices related to routing tiles up to $MF_{RoutingTiles} = R \times (4k)^2$.

The problem can then be re-written as a function of the geometry. Considering Fig.1g, let i be an integer and $(2i+1)^2$ the total number of tiles. The number of neuron tiles can be written as $T = (i+1)^2$, as we consider the case where neuron tiles form the outer ring of tiles. As a consequence, the number of routing tiles is $R = (2i+1)^2 - (i+1)^2$. Substituting such values in the previous evaluations of $MF_{NeuronTiles} + MF_{RoutingTiles}$ and remembering that $k < N \times T$, we can impose that $MF_{Mosaic} = MF_{NeuronTiles} + MF_{RoutingTiles} < MF_{MF_{ref}}$. This results in the following expression:

$$MF_{Mosaic} = MF_{NeuronTiles} + MF_{RoutingTiles} < MF_{reference} \quad (1)$$

$$(i+1)^2 4 \times k^2 + [(2i+1)^2 - (i+1)^2]((4k)^2) < (k(i+1)^2)^2 \quad (2)$$

This expression can then be evaluated for i, given a network size, giving rise to the relationships as plotted in Fig.1g in the main text.

Data availability

The MIT-BIH ECG dataset[49], the Spiking Heidelberg Datasets[71], and half-cheetah from OpenAI gym[52] are publicly accessible. All other measured data are freely available upon request.

Code availability

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

Acknowledgements

We acknowledge funding support from the H2020 MeM-Scales project (871371), Marie Skłodowska-Curie grant agreement No 861153, as well as the French ANR via Carnot funding.

Author contributions

T.D, G.I, E.V and M.P developed the Mosaic concept. T.D. and M.P. designed and laid out the circuits for fabrication. F.M. and A.P. performed the characterizations and verifications on the fabricated circuits. Y.D., T.D., F.M and M.P. developed the Mosaic simulations. All authors contributed to writing of the manuscript.

References

1. Sterling, P. Design of neurons. In *Principles of Neural Design*, 155–194, DOI: 10.7551/mitpress/9780262028707.003.0007 (The MIT Press, 2015).

2. Watts, D. J. & Strogatz, S. H. Collective dynamics of 'small-world'networks. *Nature* **393**, 440–442 (1998).

3. Kawai, Y., Park, J. & Asada, M. A small-world topology enhances the echo state property and signal propagation in reservoir computing. *Neural Networks* **112**, 15–23, DOI: 10.1016/j.neunet.2019.01.002 (2019).

4. Loeffler, A. *et al.* Topological properties of neuromorphic nanowire networks. *Front. Neurosci.* **14**, 184 (2020).

5. Park, H.-J. & Friston, K. Structural and functional brain networks: From connections to cognition. *Science* **342**, DOI: 10.1126/science.1238411 (2013).

6. Gallos, L. K., Makse, H. A. & Sigman, M. A small world of weak ties provides optimal global integration of self-similar modules in functional brain networks. *Proc. Natl. Acad. Sci.* **109**, 2825–2830, DOI: 10.1073/pnas.1106612109 (2012).

7. Sporns, O. & Zwi, J. D. The small world of the cerebral cortex. *Neuroinformatics* **2**, 145–162, DOI: 10.1385/ni:2:2:145 (2004).

8. Bullmore, E. & Sporns, O. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nat. Rev. Neurosci.* **10**, 186–198 (2009).

9. Jo, S. H. *et al.* Nanoscale memristor device as synapse in neuromorphic systems. *Nano letters* **10**, 1297–1301 (2010).

10. Ielmini, D. & Waser, R. *Resistive Switching: From Fundamentals of Nanoionic Redox Processes to Memristive Device Applications* (John Wiley & Sons, 2015).

11. Serb, A. *et al.* Unsupervised learning in probabilistic neural networks with multi-state metal-oxide memristive synapses. *Nat. Commun.* **7**, 12611 (2016).

12. Li, C. *et al.* Efficient and self-adaptive in-situ learning in multilayer memristor neural network. *Nat. Commun.* **9**, 1–8, DOI: 10.1038/s41467-018-04484-2 (2018).

13. Strukov, D., Indiveri, G., Grollier, J. & Fusi, S. Building brain-inspired computing. *Nat. Commun.* **10**, DOI: 10.1038/s41467-019-12521-x (2019).

14. Kingra, S. K. *et al.* SLIM: Simultaneous Logic-In-Memory computing exploiting bilayer analog OxRAM devices. *Sci. Reports* **10**, 1–14 (2020).

15. 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).

16. Woźniak, S., Pantazi, A., Bohnstingl, T. & Eleftheriou, E. Deep learning incorporating biologically inspired neural dynamics and in-memory computing. *Nat. Mach. Intell.* **2**, 325–336 (2020).

17. Sebastian, A., Le Gallo, M., Khaddam-Aljameh, R. & Eleftheriou, E. Memory devices and applications for in-memory computing. *Nat. Nanotechnol.* **15**, 529–544 (2020).

18. 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).

19. Jouppi, N. P. *et al.* In-datacenter performance analysis of a Tensor Processing Unit. In *Proceedings of the 44th annual international symposium on computer architecture*, 1–12 (2017).

20. Yu, S., Sun, X., Peng, X. & Huang, S. Compute-in-memory with emerging nonvolatile-memories: challenges and prospects. In *2020 IEEE Custom Integrated Circuits Conference (CICC)*, 1–4 (IEEE, 2020).

21. Joksas, D. *et al.* Committee machines—a universal method to deal with non-idealities in memristor-based neural networks. *Nat. Commun.* **11**, 1–10 (2020).

22. Zidan, M. A., Strachan, J. P. & Lu, W. D. The future of electronics based on memristive systems. *Nat. Electron.* **1**, 22–29 (2018).

23. 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).

24. Yao, P. *et al.* Fully hardware-implemented memristor convolutional neural network. *Nature* **577**, 641–646 (2020).

25. Bullmore, E. & Sporns, O. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nat. Rev. Neurosci.* **10**, 186–198 (2009).

26. Duan, S., Hu, X., Dong, Z., Wang, L. & Mazumder, P. Memristor-based cellular nonlinear/neural network: design, analysis, and applications. *IEEE Transactions on Neural Networks Learn. Syst.* **26**, 1202–1213 (2014).

27. Ascoli, A., Messaris, I., Tetzlaff, R. & Chua, L. O. Theoretical foundations of memristor cellular nonlinear networks: Stability analysis with dynamic memristors. *IEEE Transactions on Circuits Syst. I: Regul. Pap.* **67**, 1389–1401 (2019).

28. Wang, R. *et al.* Implementing in-situ self-organizing maps with memristor crossbar arrays for data mining and optimization. *Nat. Commun.* **13**, 1–10 (2022).

29. Likharev, K., Mayr, A., Muckra, I. & Türel, Ö. Crossnets: High-performance neuromorphic architectures for cmol circuits. *Annals New York Acad. Sci.* **1006**, 146–163 (2003).

30. Betta, G., Graffi, S., Kovacs, Z. M. & Masetti, G. Cmos implementation of an analogically programmable cellular neural network. *IEEE Transactions on Circuits Syst. II: Analog. Digit. Signal Process.* **40**, 206–215 (1993).

31. Khacef, L., Rodriguez, L. & Miramond, B. Brain-inspired self-organization with cellular neuromorphic computing for multimodal unsupervised learning. *Electronics* **9**, 1605 (2020).

32. Lin, P., Pi, S. & Xia, Q. 3d integration of planar crossbar memristive devices with cmos substrate. *Nanotechnology* **25**, 405202 (2014).

33. Boahen, K., Nomura, M., Vidal, E. R. & Rullen, R. V. Address-event senders and receivers: Implementing direction-selectivity and orientation-tuning (1998).

34. Moradi, S., Qiao, N., Stefanini, F. & Indiveri, G. A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs). *Biomed. Circuits Syst. IEEE Transactions on* **12**, 106–122, DOI: 10.1109/TBCAS.2017.2759700 (2018).

35. Park, J., Yu, T., Joshi, S., Maier, C. & Cauwenberghs, G. Hierarchical address event routing for reconfigurable large-scale neuromorphic systems. *IEEE transactions on neural networks learning systems* **28**, 2408–2422 (2016).

36. Indiveri, G. *et al.* Neuromorphic silicon neuron circuits. *Front. Neurosci.* **5**, 1–23, DOI: 10.3389/fnins.2011.00073 (2011).

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

38. Cai, F. *et al.* Power-efficient combinatorial optimization using intrinsic noise in memristor hopfield neural networks. *Nat. Electron.* **3**, 409–418 (2020).

39. Bartolozzi, C. & Indiveri, G. Synaptic dynamics in analog vlsi. *Neural computation* **19**, 2581–2603 (2007).

40. Kung, H. T. & Leiserson, C. E. Systolic arrays for VLSI. Tech. Rep., Carnegi-Mellon Univ. Pittsburg PA (1978).

41. Grossi, A. *et al.* Fundamental variability limits of filament-based RRAM. In *2016 IEEE International Electron Devices Meeting (IEDM)*, 4.7.1–4.7.4, DOI: 10.1109/IEDM.2016.7838348 (2016).

42. Werbos, P. J. Backpropagation through time: What it does and how to do it. *Proc. IEEE* **78**, 1550–1560 (1990).

43. Neftci, E. O., Mostafa, H. & Zenke, F. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Process. Mag.* **36**, 51–63 (2019).

44. Esmanhotto, E. *et al.* High-density 3D monolithically integrated multiple 1T1R multi-level-cell for neural networks. In *2020 IEEE International Electron Devices Meeting (IEDM)*, 36–5 (IEEE, 2020).

45. Dalgaty, T. *et al.* In situ learning using intrinsic memristor variability via Markov chain Monte Carlo sampling. *Nat. Electron.* **4**, 151–161 (2021).

46. Zhao, M. *et al.* Investigation of statistical retention of filamentary analog rram for neuromophic computing. In *2017 IEEE International Electron Devices Meeting (IEDM)*, 39.4.1–39.4.4, DOI: 10.1109/IEDM.2017.8268522 (2017).

47. Moro, F. *et al.* Hardware calibrated learning to compensate heterogeneity in analog rram-based spiking neural networks. *IEEE Int. Symp. Circuits Syst.* (2022).

48. Cramer, B., Stradmann, Y., Schemmel, J. & Zenke, F. The heidelberg spiking data sets for the systematic evaluation of spiking neural networks. *IEEE Transactions on Neural Networks Learn. Syst.* (2020).

49. Moody, G. B. & Mark, R. G. The impact of the MIT-BIH arrhythmia database. *IEEE Eng. Medicine Biol. Mag.* **20**, 45–50 (2001).

50. Lee, H.-Y., Hsu, C.-M., Huang, S.-C., Shih, Y.-W. & Luo, C.-H. Designing low power of sigma delta modulator for biomedical application. *Biomed. Eng. Appl. Basis Commun.* **17**, 181–185 (2005).

51. Corradi, F. & Indiveri, G. A neuromorphic event-based neural recording system for smart brain-machine-interfaces. *IEEE Transactions on Biomed. Circuits Syst.* **9**, 699–709 (2015).

52. Brockman, G. *et al.* OpenAI Gym (2016). arXiv:1606.01540.

53. Luo, W. *et al.* End-to-end active object tracking and its real-world deployment via reinforcement learning. *IEEE Transactions on Pattern Analysis Mach. Intell.* **42**, 1317–1332 (2020).

54. Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V. & Hutter, M. Learning quadrupedal locomotion over challenging terrain. *Sci. Robotics* **5**, DOI: 10.1126/scirobotics.abc5986 (2020).

55. Salimans, T., Ho, J., Chen, X., Sidor, S. & Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv* (2017).

56. Vinyals, O. *et al.* Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **575**, 350–354, DOI: 10.1038/s41586-019-1724-z (2019).

57. OpenAI *et al.* Learning Dexterous In-Hand Manipulation. *arXiv:1808.00177 [cs, stat]* (2019).

58. Jordan, J., Schmidt, M., Senn, W. & Petrovici, M. A. Evolving interpretable plasticity for spiking networks. *eLife* **10**, e66273, DOI: 10.7554/eLife.66273 (2021).

59. Merolla, P., Arthur, J., Alvarez, R., Bussat, J.-M. & Boahen, K. A multicast tree router for multichip neuromorphic systems. *Circuits Syst. I: Regul. Pap. IEEE Transactions on* **61**, 820–833, DOI: 10.1109/TCSI.2013.2284184 (2014).

60. Painkras, E. *et al.* SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation. *IEEE J. Solid-State Circuits* **48**, 1943–1953, DOI: 10.1109/JSSC.2013.2259038 (2013).

61. Benjamin, B. V. *et al.* Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proc. IEEE* **102**, 699–716 (2014).

62. Davies, M. *et al.* Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* **38**, 82–99 (2018).

63. Rabaey, J. M., Chandrakasan, A. P. & Nikolić, B. *Digital integrated circuits: a design perspective*, vol. 7 (Pearson education Upper Saddle River, NJ, 2003).

64. Davies, M. *et al.* Advancing neuromorphic computing with Loihi: A survey of results and outlook. *Proc. IEEE* **109**, 911–934 (2021).

65. Chicca, E., Stefanini, F., Bartolozzi, C. & Indiveri, G. Neuromorphic electronic circuits for building autonomous cognitive systems. *Proc. IEEE* **102**, 1367–1388, DOI: https://doi.org/10.1109/JPROC.2014.2313954 (2014).

66. Dalgaty, T., Payvand, M., De Salvo, B. *et al.* Hybrid CMOS-RRAM neurons with intrinsic plasticity. In *IEEE ISCAS*, 1–5 (IEEE, 2019).

67. Joshi, V. *et al.* Accurate deep neural network inference using computational phase-change memory. *Nat. Commun.* **11**, DOI: 10.1038/s41467-020-16108-9 (2020).

68. Wan, W. *et al.* A compute-in-memory chip based on resistive random-access memory. *Nature* **608**, 504–512 (2022).

69. Corradi, F., Bontrager, D. & Indiveri, G. Toward neuromorphic intelligent brain-machine interfaces: An event-based neural recording and processing system. In *Biomedical Circuits and Systems Conference (BioCAS)*, 584–587, DOI: 10.1109/BioCAS.2014.6981793 (IEEE, 2014).

70. Freeman, C. D. *et al.* Brax - a differentiable physics engine for large scale rigid body simulation (2021).

71. Cramer, B., Stradmann, Y., Schemmel, J. & Zenke, F. The heidelberg spiking data sets for the systematic evaluation of spiking neural networks. *IEEE Transactions on Neural Networks Learn. Syst.* (2020).