

Supplementary Material: Single Neuromorphic Memristor closely emulates Multiple Synaptic Mechanisms for Energy Efficient Neural Networks

Christoph Weilenmann^{1*}, Alexandros Nikolaos Ziogas¹,
Till Zellweger¹, Kevin Portner¹, Marko Mladenović¹,
Manasa Kaniselvan¹, Timoleon Moraitis², Mathieu Luisier¹,
Alexandros Emboras¹

¹Integrated Systems Laboratory, ETH Zurich, Zurich, Switzerland.

²Noemon AG, Zurich, Switzerland.

*Corresponding author(s). E-mail(s): weilenmc@iis.ee.ethz.ch;

S1 Fabrication Process

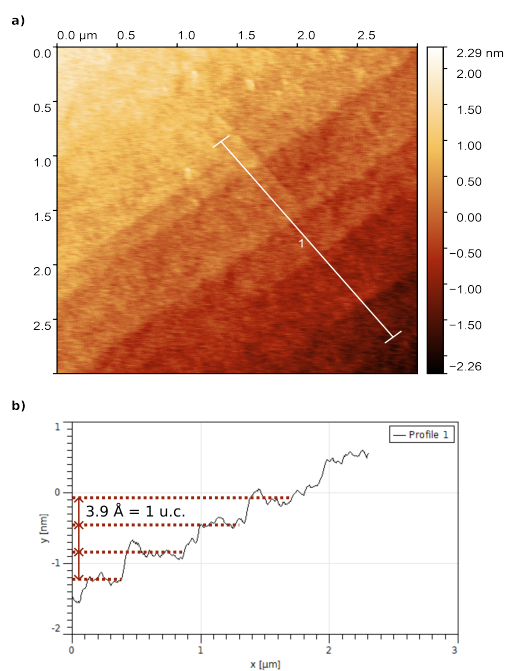


Fig. S1 a) AFM image of STO surface and b) height profile through the line displayed in a). Terraces of $3.9 \text{ \AA}$ height are observed, which corresponds to 1 unit cell (u.c.) of STO.

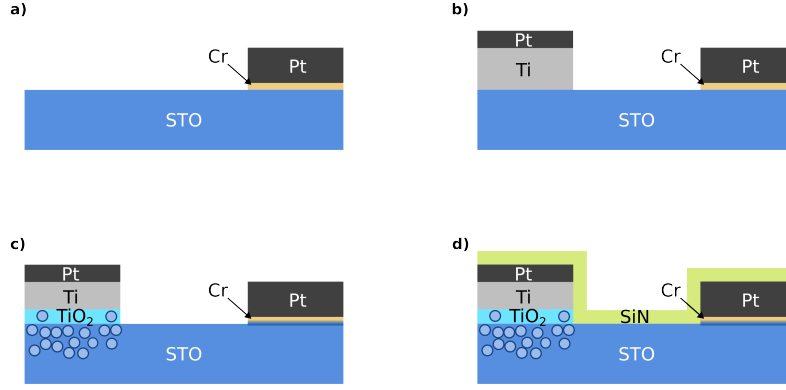


Fig. S2 Fabrication process as detailed in the Method Section of the main text. **a)** Deposition of Pt electrode with a Cr adhesion layer on the STO substrate. **b)** Deposition of Ti electrode with Pt capping layer. **c)** Annealing at 300°C in Ar atmosphere, creating a thermal oxide (TiO₂) and oxygen vacancies (blue circles) at the STO/Ti electrode interface. The Cr adhesion layer diffuses into the STO substrate (dark blue gradient at the Cr layer). **d)** Deposition of an SiN encapsulation layer.

S2 Filamentary vs Non-Filamentary Switching

Fig. S3a shows a typical non-filamentary Conductance-Voltage characteristic. The switching direction is counter clockwise (SET for positive voltages, RESET for negative voltages). The SET and RESET operations are smoothly varying without any sudden jumps in conductance. The conductance is in the nano Siemens range. Fig. S3b by contrast shows the filamentary switching regime. The different colored curves correspond to subsequent measurements, starting with the blue curve. By applying a large enough positive voltage (roughly 4.5V for this device) a sudden jump in conductance is observed (forming). The subsequent measurements obey a clock-wise switching direction (SET for negative voltages, RESET for positive voltages). The conductance is also much higher than for the non-filamentary case. The voltage was applied to the Pt electrode and the Ti electrode was grounded.

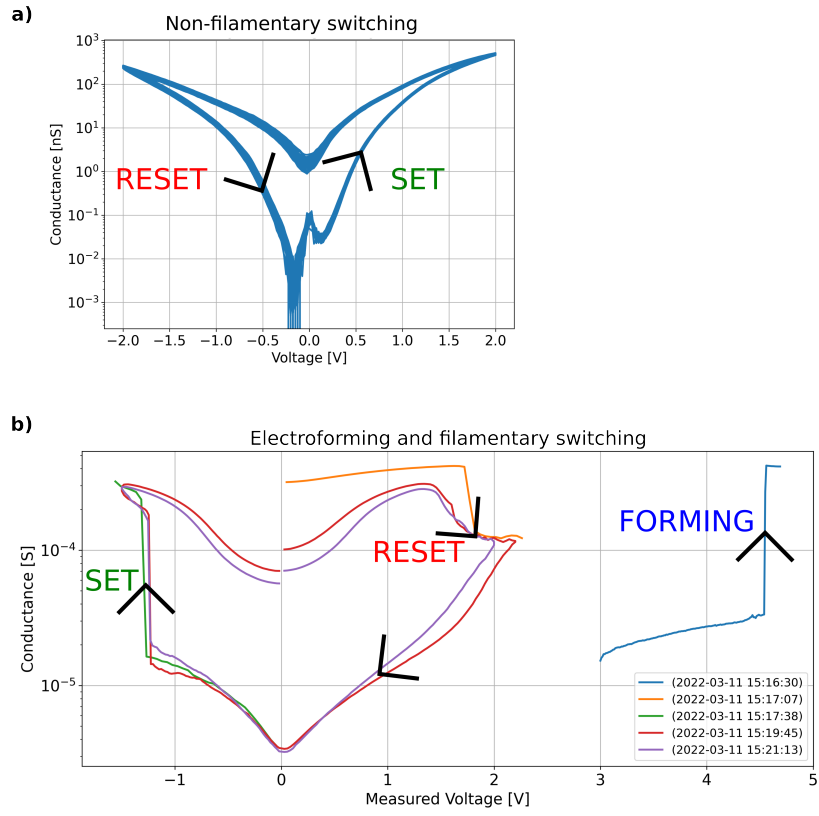


Fig. S3 a) Non-filamentary switching. b) Electroforming and filamentary switching.

S3 Retention characteristics after SET pulses

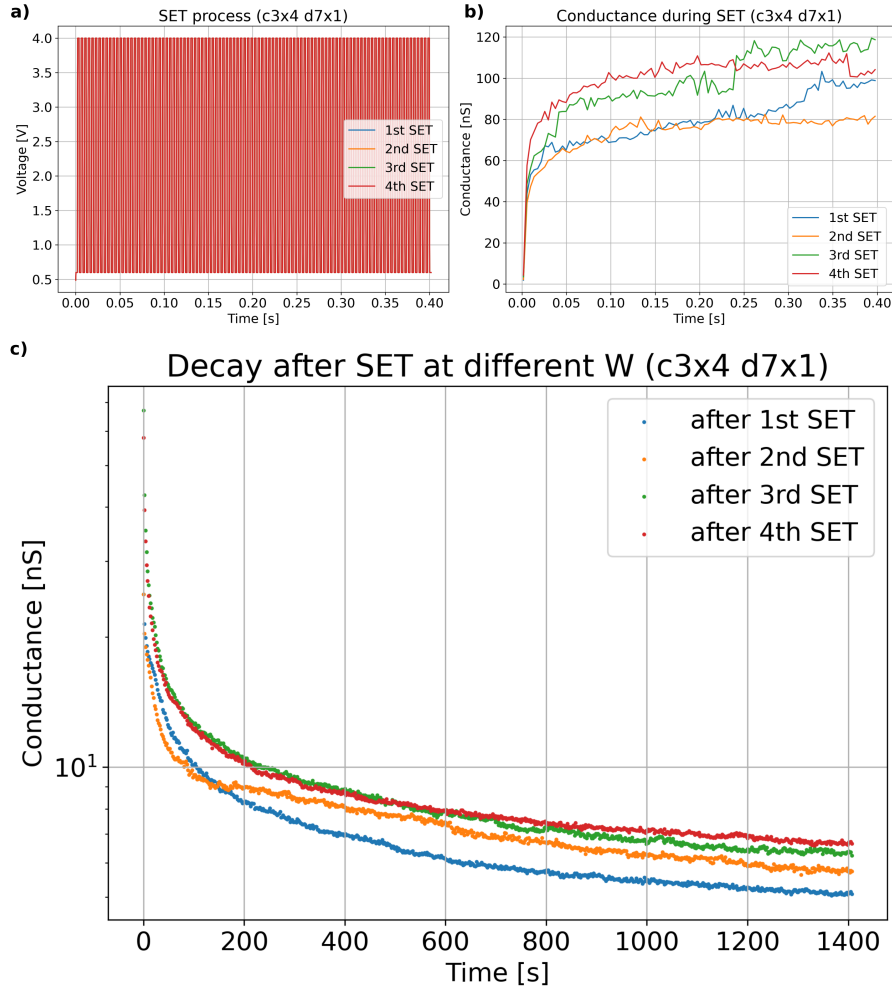


Fig. S4 Retention characteristic after four subsequent SET procedures. **a)** Applied voltage pulses during the 4 SET procedures. The pulse width is 2ms. **b)** Conductance change during SET procedure (measured inbetween SET pulses at 0.6V read voltage. **c)** Retention characteristics after SET process. The conductance was read out with voltage pulses of 0.6V / 90ms and a period of 2s. The bias inbetween the read pulses was 0V.

S4 Endurance

We observe a stable switching behavior for the first 150 cycles, where the conductance dynamics vary little from pulse to pulse. After this initial phase the device's behavior becomes more unstable. This evolution might be due to the influence of the atmosphere on the endurance of STO devices [1]. The functionality of the device is however still intact since the short-term pulses still induce a similar conductance change as before the decrease in device conductance. As a matter of fact, we have been conducting measurements on this particular device for the past two years. We have put the

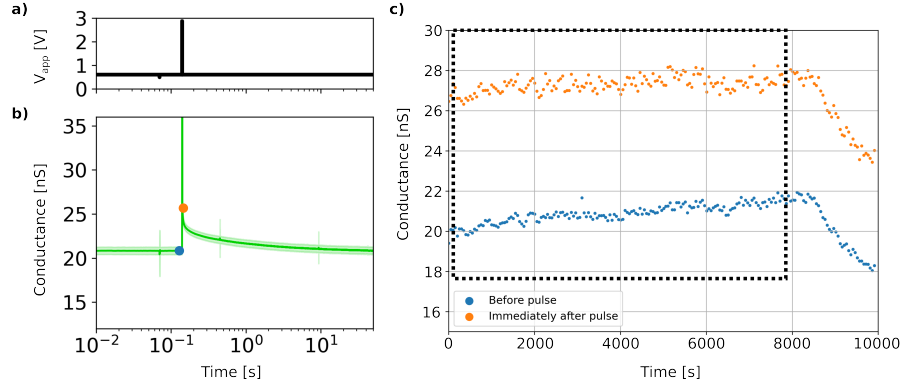


Fig. S5 **a)** Endurance measurement protocol comprising 1 of the 200 3V/100 μ s pulses with a period of 50s. The conductance was read out with a constant 0.6V read voltage to capture the short-term decay dynamics. **b)** Mean and standard deviation of the conductance for the first 150 cycles. The blue and orange dots denote the conductance before and after the pulse used in c). **c)** Shows the conductance immediately before (blue dots) and after (orange dots) each of the 200 pulses. The first 150 cycles are marked by the dotted rectangle.

device through many switching cycles and various experimental protocols, with each experiment lasting several minutes to hours. We did not see any irreversible loss of functionality.

Please also note that while 200 cycles is not usually considered an adequate number to report on the endurance of a device it is important to keep in mind that each cycle in our case lasts 50s. This relatively long duration can be attributed to the short-term decay of the device, which happens on this timescale. In conventional endurance measurements, in contrast, two stable resistance states are considered that can be switched at high frequencies, which allows the measurement of a large number of cycles in a short time. A measurement time of 10'000 seconds is often reported, which is what informed our own choice of running the measurement for 10'000 seconds (200 cycles x 50s).

S5 Device to Device variation

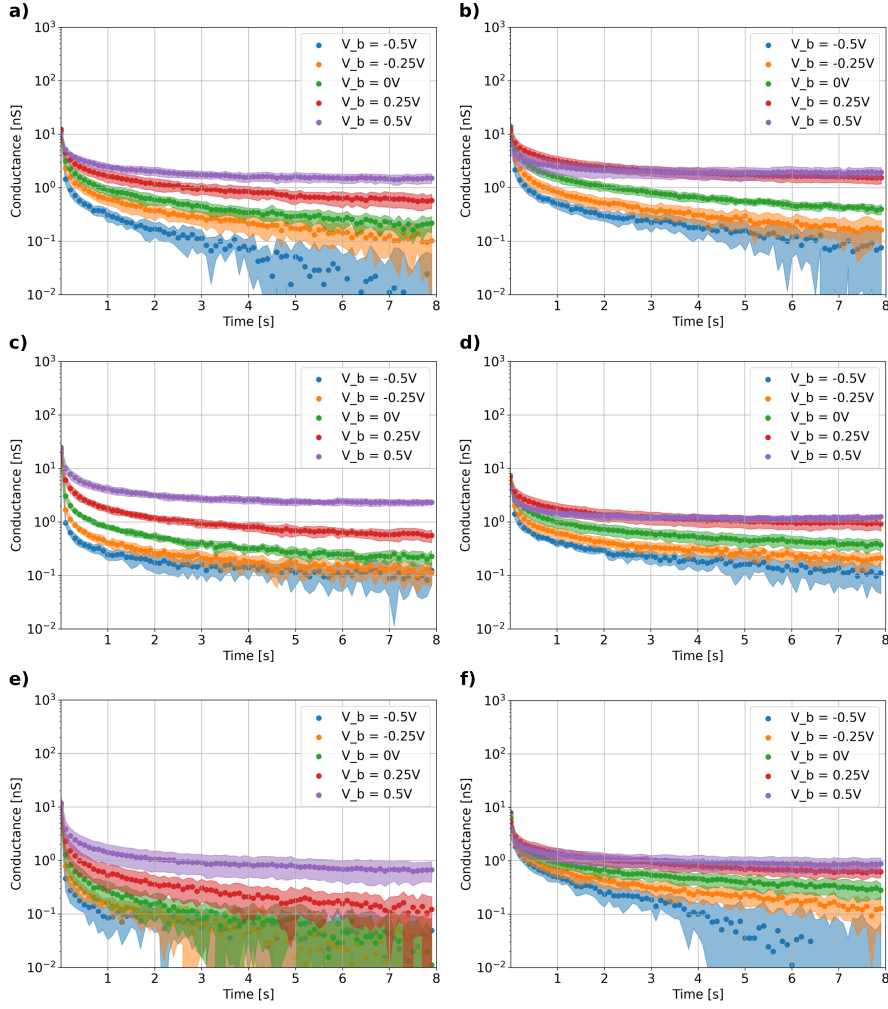


Fig. S6 a) - f) Measurement of 6 different devices showing the controllability of the short-term decay time for 5 different values of V_b (bias voltage applied between two consecutive read pulses). A 2V/10ms voltage pulse is applied at $t=0$ and the conductance is then periodically read out with 0.5V/10ms read pulses. At each read pulse the mean conductance value over 9 cycles (dots in figures) is measured. The shaded area represents the corresponding standard deviations.

We observe that in all devices the decay time can be reliably adjusted, as in Fig. 3d of the main text, but the conductance values vary from one device to the other. The origin of this variability can be traced back to the employed planar structure, where the electrode geometry and gap size between the contacts cannot be precisely controlled.

S6 Physical mechanism summary

Function ☐ Memory ☐ Comp.		Physical Mechanism
1)	weight: long-term (W)	The long-term weight (W) is encoded by the conductance. The dynamics are long-term, because the V_O's are far away from the Pt contact , where their mobility is low (activation energy for oxygen migration in bulk STO $E_A^{(bulk)} = 0.64\text{eV}$)
2)	input-weight mult.	The input (U) and total weight (G) are encoded via the pulse voltage and the device conductance respectively. The multiplication is computed through Ohm's law: $I = G \cdot U$
3)	plasticity: long-term	The setting of the long-term weight (W) is achieved by applying 100 voltage pulses (4V / 500us). The high voltage combined with the long pulse train leads to V _O generation at the Pt contact and migration far away from the interface .
4)	weight: short-term (F)	The short-term weight (F) is encoded by the conductance. The dynamics are short-term, because the V_O's are close to the Pt contact , where they can be filled by O in the Pt (act. energy for oxygen migration across Pt/STO interface $E_A^{(inter.)} = 0.16 \text{ to } 0.86\text{eV}$)
5)	plasticity: short-term (ΔF)	The setting of the short-term weight (F) is achieved by applying a single voltage pulse (2 to 3V / 20 to 500us). The short pulse duration leads to a limited migration away from the interface, so the generated V_O's end up close to the Pt contact .
6)	metaplasticity: control decay time	Metaplasticity is implemented by increasing or decreasing the flux of VO migration back to the Pt contact. By applying a negative voltage to the Pt contact the flux is increased and by applying a positive voltage the flux is decreased.

Fig. S7 Table summarizing the underlying physical mechanism of each synaptic function. The value for the activation energy of bulk migration was taken from [2] and the activation energy for oxygen migration across the Pt/STO interface from [3].

Another relevant mechanism might be protonic conduction, which was shown to be significant in acceptor-doped STO [4, 5]. This would affect the device conductance, as reported for the high resistance state of STO memristors under humid atmospheres [1].

S7 Energy Calculation for Short-Term Weight Update

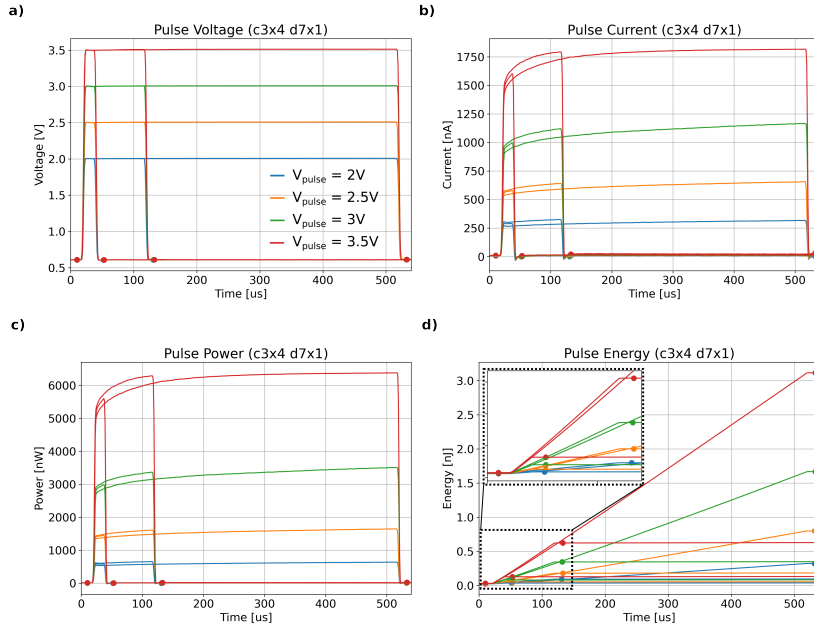


Fig. S8 a) Voltage, b) Current, c) Power and d) Energy during the application of pulses with different voltages and widths.

Fig. S8a shows the applied voltage pulses with different voltages and widths. Fig. S8b shows the measured current that result from the pulses in Fig. S8a. In Fig. S8c the power $P = U \cdot I$ is displayed. By integrating the power over time the energy as a function of the time is obtained (Fig. S8d). The total energy per pulse is then calculated by subtracting the energy consumed before the pulse from the energy after the pulse. The timestamps before and after each pulse are marked with a dot in all figures.

S8 Negative Delta F

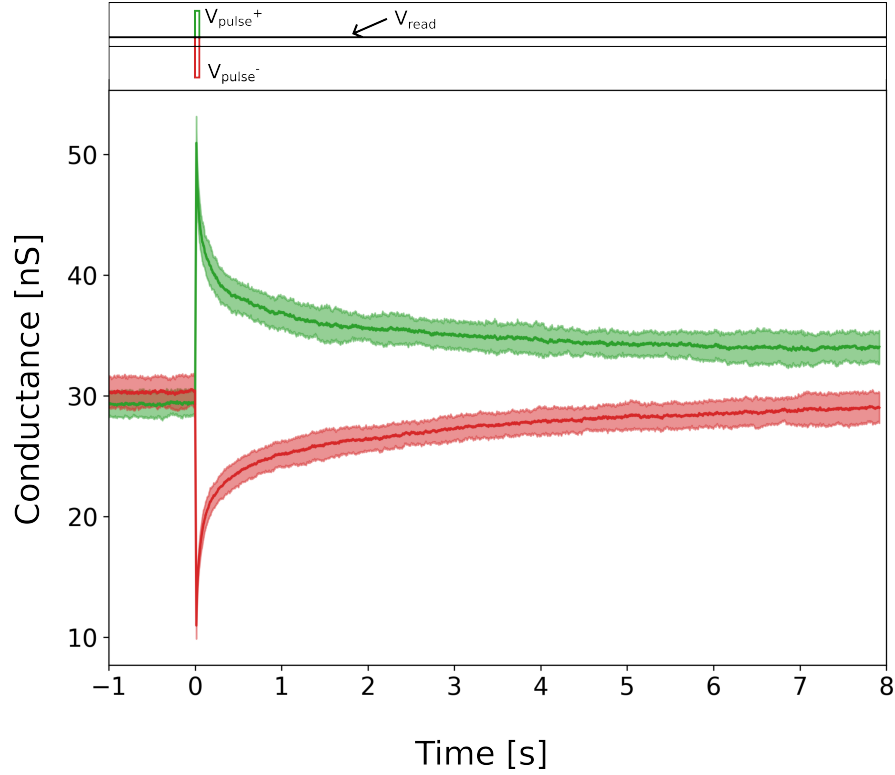


Fig. S9 Positive and negative ΔF achieved with a positive ($V_{pulse}^+ = 2V$) and negative ($V_{pulse}^- = -2V$) pulse voltage.

S9 Controllable Decay Time Constant

Fig. S10a shows the short-term dynamics of the synaptic weight in biological synapses reprinted with permission from [6]. The synaptic weight is determined in vitro by measuring the potential at the postsynaptic neuron, called field excitatory postsynaptic potential (fEPSP). When an electrical stimulus is applied through an electrode (black arrow in Fig. S10a) the synaptic weight increases and subsequently decays back. We note that the decay does not follow a simple exponential relationship as is assumed in our STPN network model. Rather the short-term weight initially decays quickly followed by a slower decrease. Fig. S10b - S10f show the raw data that was used to make Fig. 3d and 3e of the main text. The measurement protocol is given in Fig. 3d and is explained in detail in the main text. In each of the figures Fig. S10b - S10f five measurements of the conductance are plotted as a function of the time (solid lines). We can see that the measured curves overlap well. The dashed lines in each figure denote

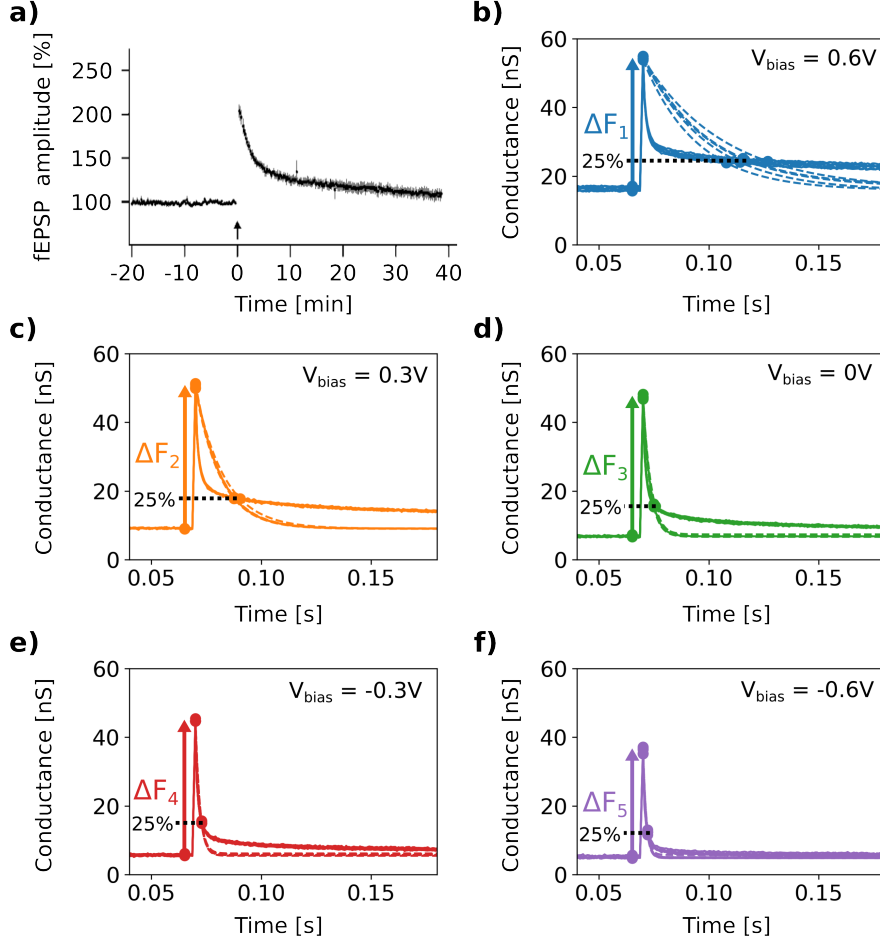


Fig. S10 a) Synaptic short-term plasticity in biological synapses reprinted with permission from [6] ©2010 by the Massachusetts Institute of Technology. All rights reserved. The field excitatory postsynaptic potential (fEPSP) is measured before ($t < 0$ min) and after ($t > 0$ min) stimulation. The fEPSP is normalized to a baseline value 5min prior to stimulation of the synapse (at $t = 0$ min). The fEPSP is a direct measure of the synaptic weight. It is observed that the decay after stimulation consists of a fast initial decrease followed by a second, slower decay. b) - f) Conductance measurements used to make Fig. 3d and 3e of the main text. Each plot shows 5 measurements overlaid on top of each other (solid lines) as well as an exponential fit for each measurement (dashed lines). The short-term weight update (ΔF) is shown as well as the chosen threshold ($25\% \cdot \Delta F$) for the calculation of the decay time constant Λ .

an exponential fit for all of the five measurements. The fit is given by the formula:

$$G = \Delta F \cdot \Lambda^{\frac{t-t_0}{t_{step}}} + G_{start} \quad (1)$$

where G is the conductance, ΔF the short term weight update, Λ the decay time constant, t the time, t_0 the time right after the pulse, $t_{step} = 7.3\text{ms}$ the simulation step

size of the neural network simulation and G_{start} the initial conductance before the pulse. This formulation is derived from the decay computations of the STPN network: At each timestep the short-term weight decays via the formula $F^{(t+1)} = \Lambda \cdot F^{(t)}$. This leads to an exponential decay: $F^{(t)} = \Delta F \cdot \Lambda^{\frac{t}{t_{step}}}$. Furthermore the total synaptic weight is composed of a long- and short-term component according to: $G = W + F^{(t)}$, where $W = G_{start}$. Therefore, the final equation for the decay is given by eq. 1. To extract the decay time constant Λ from our measurements eq. 1 is transformed to yield:

$$\Lambda = \left(\frac{G_{25\%} - G_{start}}{\Delta F} \right)^{\frac{t_{step}}{t_{25\%} - t_0}} \quad (2)$$

where $G_{25\%} = 25\% \cdot \Delta F + G_{start}$ and $t_{25\%}$ is the time when the conductance decayed to the value of $G_{25\%}$, i.e. one forth of the conductance at the start of the decay. The threshold of 25% was chosen arbitrarily. The exponential decay model was selected for its simplicity and to adopt the convention used in the wider short-term plasticity literature [7, 8]. However we see for example in Fig. S10b that the exponential model is not a good fit for the actual behavior in our devices, especially for large bias voltages. Interestingly the decay in our devices more closely mimics the behavior of biological synapses than the simplified exponential model, albeit at smaller timescales (s instead of minutes), as can be seen by comparing the measurements in figures Fig. S10b - S10f to the results in Fig. S10a. We leave the investigation of the impact of these more bio-realistic decay dynamics on the performance of our network to future work.

S10 Modified STPN Model

The equations describing the forward pass through an STPN layer are reproduced from [9]:

$$\mathbf{G}^{(t)} = \mathbf{W} + \mathbf{F}^{(t)} \quad (3)$$

$$\mathbf{h}^{(t)} = \tanh(\mathbf{G}^{(t)} \mathbf{x}^{(t)}) \quad (4)$$

$$\mathbf{F}^{(t+1)} = \underbrace{\mathbf{\Gamma} \odot (\mathbf{x}^{(t)} \otimes \mathbf{h}^{(t)})}_{\Delta F} + \underbrace{\mathbf{\Lambda} \odot \mathbf{F}^{(t)}}_{\text{decay}} \quad (5)$$

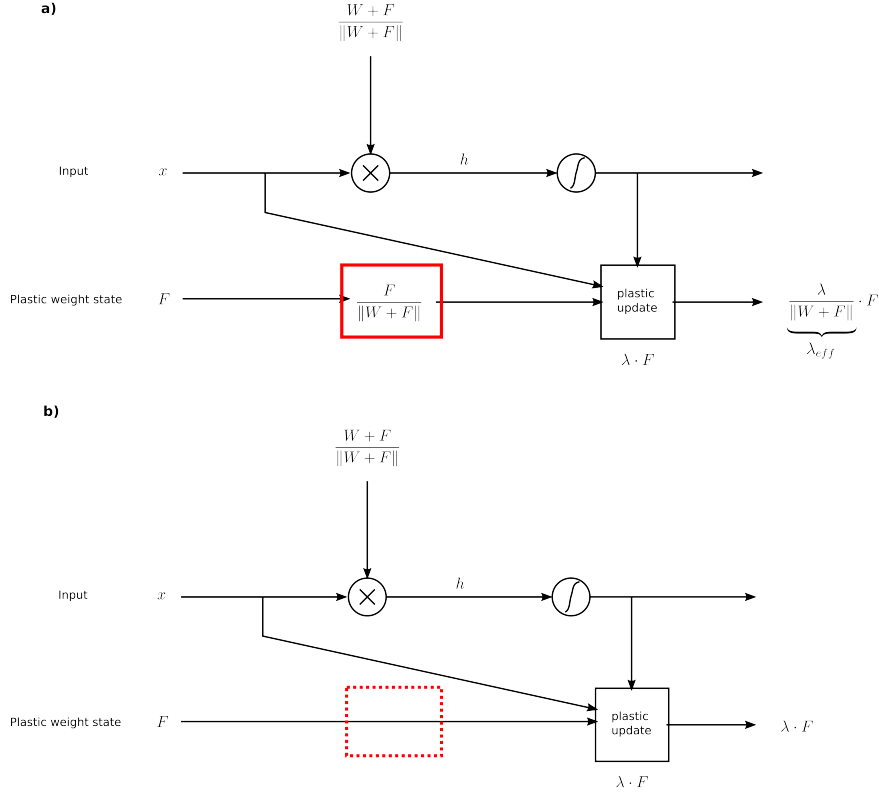


Fig. S11 **a)** Flow diagram of a forward pass in the original STPN unit. **b)** Flow diagram of a forward pass in the modified STPN unit.

S11 Non-idealities Short-Term Weight Update

The short-term weight is updated throughout the course of the game. Fig. S12a shows a histogram of the maximum of these weight updates (ΔF) for each synapse, without the addition of any non-idealities. We can see that most of the synapses have a maximum absolute value of $|\Delta F| < 20$, with only a few synapses exhibiting maximum absolute values above 20. The synapse with the highest absolute value exhibits a ΔF of roughly 60. In order to implement the short-term weight updates on our devices we have to 1) limit the maximum value of ΔF as well as 2) make sure that the short-term weight can be updated in discrete steps. This is because 1) there is a limit to the voltage that can be safely applied to the device and 2) every device exhibits some noise, which prevents an arbitrary precision in the weight update. For these reasons the two non-idealities above were implemented into the simulation. In Fig. S12b we can see the same histogram as in Fig. S12a, but with the non-idealities implemented. The highest absolute values are limited to 20 and the minimum update step is 0.5. Both implementations achieved the maximum reward of 21 at the end of the game.

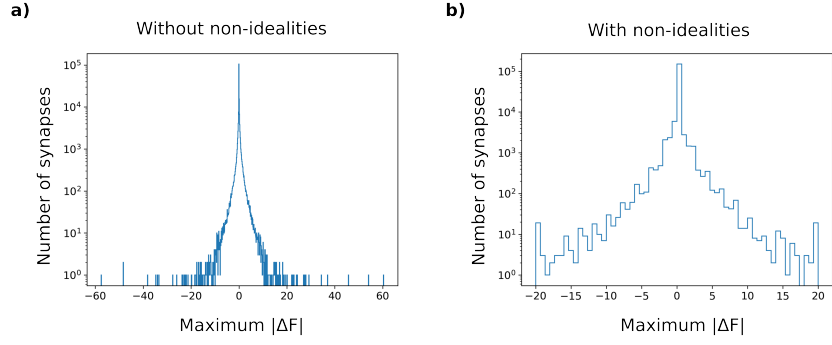


Fig. S12 Histogram of the ΔF with the largest absolute value for each synapse. **a)** Without non-idealities, **b)** with non-idealities.

S12 Maximum reward during training

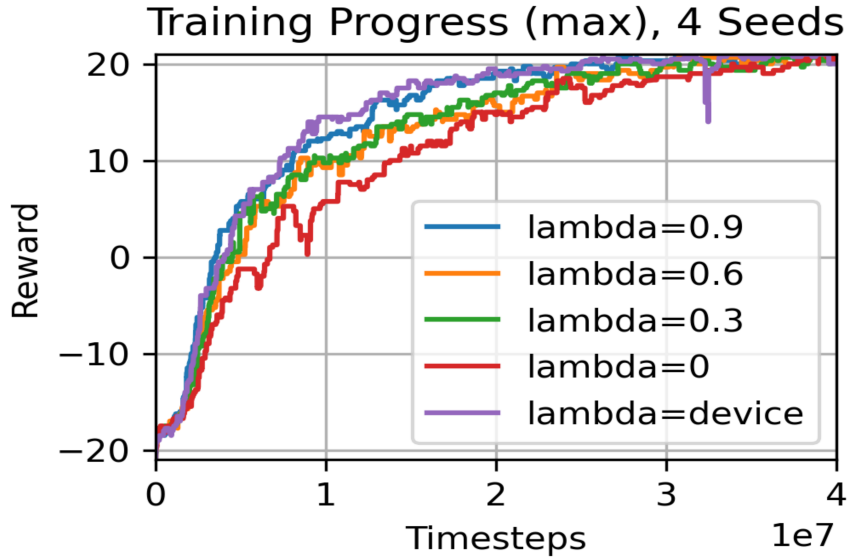


Fig. S13 Maximum reward of 16 agents during training for five different ranges of Λ . The value for λ in the legend denotes the upper limit of Λ with the lower limit fixed to 0 or in the "device" case to the range $\Lambda = [0.08, 0.92]$.

The reward that the best agent earns during training reaches the maximum of 21 in $4e7$ timesteps for all ranges of Λ .

S13 Mean reward during training including non-volatile cases

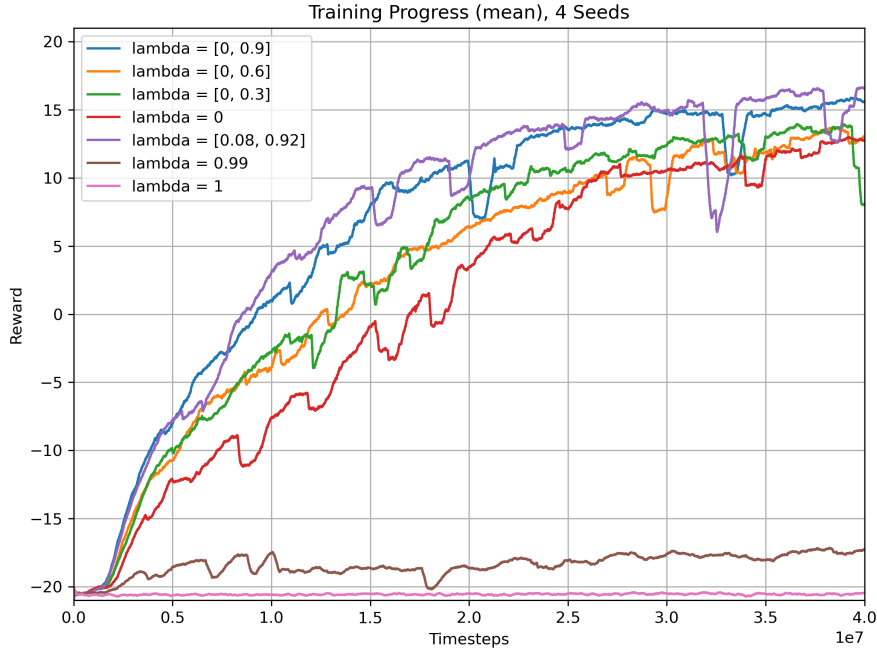


Fig. S14 Same figure as Fig. 4c) in the main text with the two additional cases $\Lambda = 0.99$ and $\Lambda = 1$

Λ can either be learned during training (cases $\lambda = [0, 0.3]$, $\lambda = [0, 0.6]$, $\lambda = [0, 0.9]$ and $\lambda = [0.08, 0.92]$) or fixed at the beginning of the training procedure ($\lambda = 0$, $\lambda = 0.99$ and $\lambda = 1$). It is observed that for fixed Λ only the $\lambda = 0$ case results in successful training (albeit not as fast or robust as the cases where Λ is learned). The two cases that correspond to non-volatile behavior ($\lambda = 0.99$ and $\lambda = 1$) fail to learn the task. This shows that non-volatile memristors would not be suitable as STPN synapses.

S14 Training with unconstrained lambda

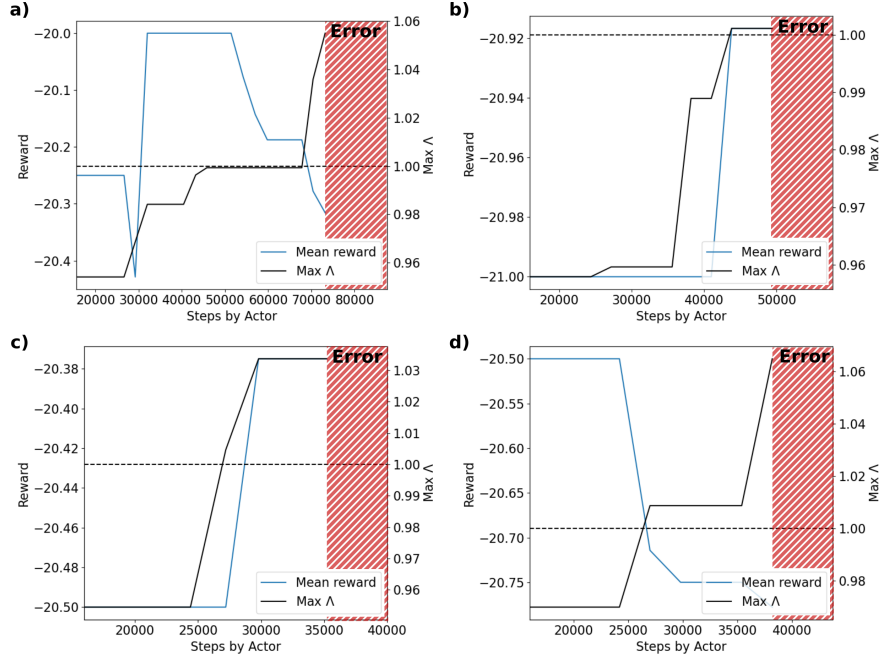


Fig. S15 Four training runs with different random seeds **a) - d)** where the values that Λ can take are not constrained. The blue curve shows the mean reward and the black curve shows the maximum value of Λ , which both change during training. The dotted line denotes $\Lambda = 1$. A short time after the maximum of lambda exceeds 1 the training algorithm throws an error as indicated by the red shaded region.

S15 Operations during Atari Pong Simulation

The simulation of an Atari Pong game, as described in the main text is divided into steps that are repeated until the game ends. During each of these steps first an action is computed by the network based on the observation of the environment (i.e., Atari Pong), which is then simulated in the game. Figure S16a shows the function calls (colored bars) during one such step, where the length of the bars represents the duration of each function. The latency of each step is on average 7.3ms. Less than half of this time is spent on the computation of the forward pass through the STPN network, which is marked by the dotted rectangle and is enlarged in Fig. S16b. The rest of the time is used to execute support routines of the employed reinforcement learning library (Rllib) or to simulate the environment. The GPU kernels that compute the arithmetic operations of the STPN layer are shown in Fig. S16c and S16d for the forward pass and plastic weight update respectively. The synaptic operations that can be done intrinsically in our memristors are marked by black circles and labelled with

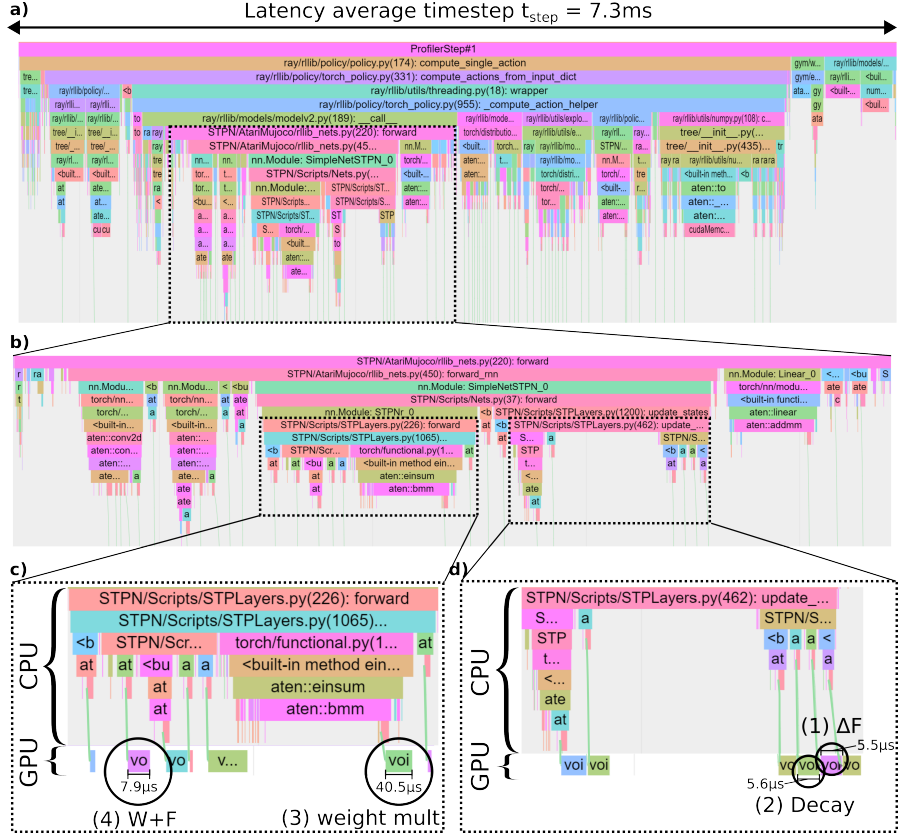


Fig. S16 a) Profiling trace showing the function calls during a single timestep of the Atari Pong simulation. Each colored bar corresponds to a function with the width of each bar signifying its duration. b) Zoom in of the forward pass through the entire STPN network. c) Zoom in of the forward pass through the STPN layer. Both CPU and GPU streams are shown and the operations that can be performed on our memristors (synaptic operations) are labelled with their latency. d) Zoom in of the state update (i.e. the short-term update of the plastic weights).

their measured GPU latency, taken from the "GPU fp16 (standard)" row of Tab. S1). Our multi-functional memristors are able to intrinsically compute the four synaptic operations (ΔF , Decay, weight mult. and W+F) using their internal physics.

S16 Power consumption GPU

Fig. S17 shows the total energy per 169984 synapses of all the considered GPU operations (ΔF , Decay, weight mult. and W+F) for different hypothetical network sizes (i.e. number of synapses calculated as $n_{syn} = (2592 + 64) \cdot x$). The energy is scaled to the original network size of 169984 synapses ($x = 64$) according to: $E \cdot 169984/n_{syn}$, which allows a direct comparison with the total energy given in Tab. 1 of the main text. The energy per 169984 synapses is minimal for $x = 1024$ for both fp32 and fp16.

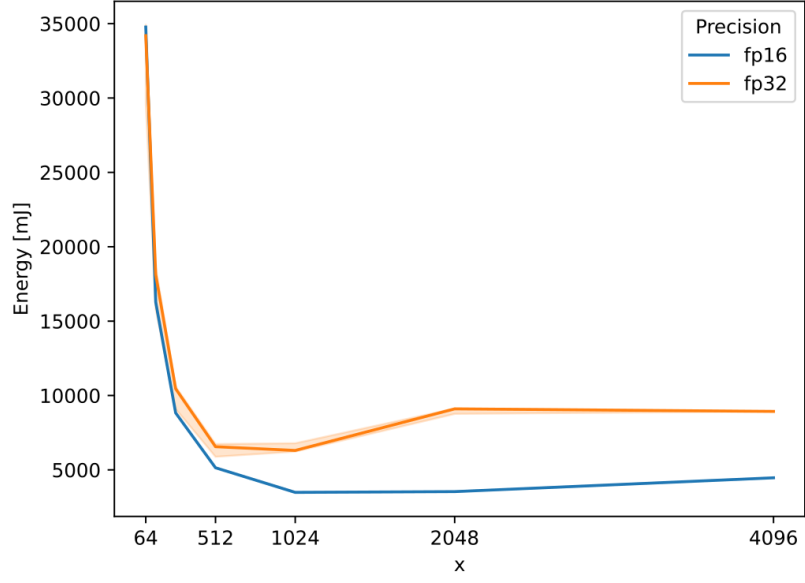


Fig. S17 Total Energy consumption of the GPU per 169984 synapses (the number of synapses in the Pong playing STPN network) as a function of the number of synapses used in the computation, which is calculated according to: $(2592 + 64) \cdot x$. The case $x = 64$ equals the energy consumption for the Pong playing STPN network. The energy per 169984 synapses is minimal for $x = 1024$. The shaded area denotes the 99% confidence interval.

This value was chosen to calculate the "GPU (optimal)" row in Tab. 1 of the main text.

S17 Latency and EDP

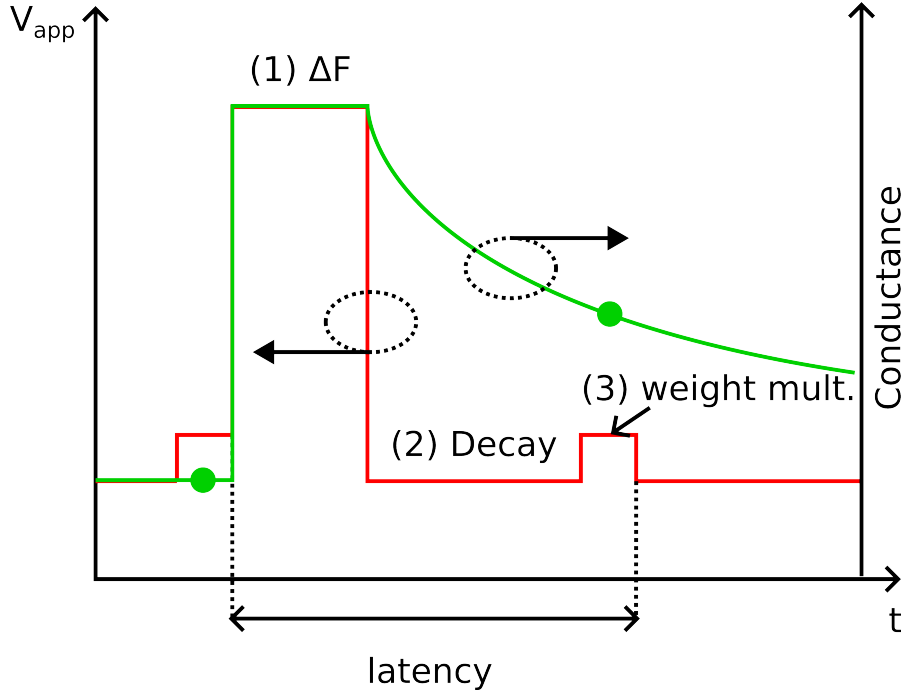


Fig. S18 Time course of a single simulation step on our memristor consisting of (1) a pulse that induces a short-term weight change (ΔF), (2) a period where the bias voltage is applied (Decay) and (3) a read pulse that performs the input-weight multiplication (weight mult.). The red curve shows the applied voltage protocol and the green curve the conductance of the memristor. The green dots denote a step of the simulation, where the conductance is read out. The duration between steps is what gives the latency for the memristor.

In the following we provide an estimation of the latency and the energy delay product (EDP) for our memristor and compare it to a GPU for the four synaptic operations:

1. ΔF : short-term update of the synaptic weight
2. Decay: performing the short-term decay
3. weight mult: reading out the conductance value and computing the multiplication between input and weight.
4. $W+F$: adding short- and long-term weight

These four operations are important during the forward pass of an STPN layer as discussed in section S15. Our multi-functional memristors are able to intrinsically compute the four synaptic operations using the physics of the device. Note that the fourth operation ($W+F$) does not have to be computed explicitly in our devices as discussed in the main text. Fig. S18 sketches the computation of a single simulation

step on our memristor by applying a voltage protocol (red) and reading out the device conductance i.e., the synaptic weight (green). A simulation step generally consists of (1) a pulse that induces a short-term weight change (ΔF), (2) a period where the bias voltage is applied (Decay) and (3) a read pulse that performs the input-weight multiplication (weight mult.) The total latency (i.e. the elapsed time per simulation step) of our device is then the sum of the latency for operations 1-3, as marked in Fig. S18. It is worth mentioning that this constitutes a worst case estimation since in the real application a synaptic weight update (ΔF) is not needed at every simulation step.

We first look at the latency of ΔF : The value of ΔF is determined by the application for each synapse and can be controlled by the width as well as the voltage of the pulse as described in Fig. 3b of the main text. The latency for this operation therefore corresponds to the width of the pulse, which can vary between 20 to 500 μs . The worst case scenario therefore is a latency of:

$$D_1^{(op)} = 500 \mu s$$

For the latency of the remainder of the simulation step we consider two cases: (1) the elapsed time between simulation steps is taken to be the one used in the energy estimation of the main text, which is given by:

$$D_{2,3}^{(op)} = D_{game}/n_{steps} = 7.4 \text{ ms}$$

where $D_{game} = 50.5 \text{ s}$ is the duration of the whole game and $n_{steps} = 6826$ the number of simulation steps. From Fig. S16a we see that most of the application runtime is made up of overhead while the actual operations that can be performed by our memristors constitute a small portion of this runtime. The latency $D_{2,3}^{(op)}$ is therefore not the minimal latency that the memristor is limited to, but is rather given by the overhead of the application. In case (2) we consider this minimal latency, which is the time between two read pulses where the bias voltage is applied. In our measurements this time is:

$$D_2^{(op,limit)} = 500 \mu s$$

and the width of a single read pulse:

$$D_3^{(op,limit)} = 200 \mu s$$

The results of this latency estimation is given in Tab. S1 under "latency" in the "Memristor" row for case (1) and in the "Memristor (limit)" row for case (2).

To compute the EDP we also require the energy for the three parts of the memristor operation. Also for the energy computation the values depend on the specifics of the application: the average energy consumed for the ΔF operation depends on the average height and width of the underlying pulses as well as on the number of pulses during a game. To account for this we consider the energy estimation of the ΔF operation of all synapses for a whole game of Atari Pong as reported in the ΔF column of Tab. 1 in the main text. The exact value without rounding is $E_1^{(tot)} = 411317.4375 \text{ nJ}$. The

estimation for the average energy consumption for a single ΔF operation is then given by:

$$E_1^{(op)} = E_1^{(tot)} / n_{steps} = 60.2575 \text{ nJ}$$

where $n_{steps} = 6826$ is again the number of simulation steps in a full game. For the decay operation we follow the same worst-case scenario as in the main text and assume a maximal decay constant Λ for all synapses, which corresponds to the maximum bias voltage of 0.6V. The read voltage for the weight mult. operation is also set to 0.6V. Based on this the energy required for these two operations for the full game is reported in Tab. 1, Decay column: $E_{2,3}^{(tot)} = 35611292 \text{ nJ}$. Again we consider the two cases of (1) an overhead limited latency and (2) the minimal latency. For case 1 the estimation of the average energy consumption for the decay and weight mult. operations is given by:

$$E_{2,3}^{(op)} = E_{2,3}^{(tot)} \cdot \frac{D_{2,3}^{(op)}}{D_{game}} = 5217.0073 \text{ nJ}$$

For case 2 the energy consumed by the decay operation is:

$$E_2^{(op,limit)} = E_{2,3}^{(tot)} \cdot \frac{D_2^{(op,limit)}}{D_{game}} = 352.54 \text{ nJ}$$

and for the weight mult. operation:

$$E_3^{(op,limit)} = E_{2,3}^{(tot)} \cdot \frac{D_3^{(op,limit)}}{D_{game}} = 141.016 \text{ nJ}$$

where $D_{game} = 50.5 \text{ s}$ is again the duration of the whole game. The result of this energy estimation is given in Tab. S1 under "energy" in the "Memristor" row for case (1) and in the "Memristor (limit)" row for case (2). Using the obtained latency and energy values we report the energy delay product (EDP) in the table under "EDP" for both cases.

We also calculated the energy, latency and EDP for the GPU with floating point 16 (fp16) precision, which we report for the four synaptic operations (ΔF , Decay, W + F and weight mult.) as in Tab. 1 of the main text. The latency was measured by taking the average of 1000 executions of each of the four operations and taking the median over 10 such averages. This approach minimizes the influence of the measurement setup procedure by making sure that the kernels run much longer than the initialization of the measurement.

We look at two cases: (1) computation for 169984 synapses, which is the size of the network investigated in this work, and (2) computation for 16×169984 synapses, which was shown to result in the smallest possible energy consumption per operation on the GPU (Fig. S17). Case 1 is reported in the "GPU fp16 (standard)" row and case 2 in the "GPU fp16 (optimal)" row. We observe similar latencies for the two cases even though the matrix sizes are $16 \times$ larger for the "optimal" case, which can be attributed to the large bandwidth of GPU's that, combined with their prefetching

capabilities, allows for reduced average latency when operating at greater network sizes [10]. The absolute energy consumption for case 2 is much larger than for case 1 due to the increased matrix size. To allow for easier comparison with the memristor cases we report the energy consumption scaled down to the original network size as in the Tab. 1 of the main text. The EDP was then calculated using the absolute latency and the scaled down energy consumption. This down-scaling is only justified in the context of a comparison with the memristor case: We assume a $16\times$ larger crossbar array that consumes $16\times$ more energy and on this hardware a hypothetical application runs with the same average energy consumption and latency per operation as the Atari Pong application. This hypothetical crossbar array is then compared to the GPU computing $16\times$ larger matrices. For the comparison between memristor and GPU it doesn't matter if we scale up the memristor energy consumption by a factor of 16 or scale down the energy consumption of the GPU by a factor of 16. The "GPU fp16 (optimal)" case does therefore not correspond to an actual hardware implementation, but solely serves as a reference to compare the memristor with the GPU for a hypothetical application with a larger network size that optimally leverages the GPU's capabilities.

The factor of improvement of a memristor over a GPU implementation for both energy and EDP is given in Fig. S19 for all discussed cases. Note that the "Memristor" case is the only one out of the four cases (both memristor and GPU cases), where we consider stand-by power draw in-between synaptic operations (due to the latency of application specific overhead). This case should therefore be viewed as highly conservative. In addition, both the "Memristor" and "Memristor (limit)" case employ worst-case assumptions such as a maximum bias voltage for all synapses and a ΔF operation at each simulation step. The resulting large spread in improvement factors (ranging from $0.7\times$ to $456\times$ for EDP) shows how sensitive these calculations are to the underlying assumptions. We therefore chose to embrace the most conservative estimate and hope that our treatment is helpful in gauging the impact of the network size and application on relevant performance metrics of our memristive system.

Table S1 Energy, latency and energy delay product (EDP) per operation on all synapses for different memristor and GPU implementations. The energy is given in nJ in the memristor case and μJ for the GPU implementations. The latency is given in μs and the EDP in nJ·s

		ΔF	Decay	W + F	ST-Hebb	w. mult.	Total
Memristor	latency [μs]	500	7399.2	$\cdot^{-1}$	7899.2	$\cdot^{-2}$	7899.2
	energy [nJ]	60.3	5217	$\cdot^{-1}$	5277.3	$\cdot^{-2}$	5277.3
	EDP [nJ·s]	0.03	38.6	$\cdot^{-1}$	41.7	$\cdot^{-2}$	41.7
Memristor (limit)	latency [μs]	500	500	$\cdot^{-1}$	1000	200	1200
	energy [nJ]	60.3	352.5	$\cdot^{-1}$	412.8	141.0	553.8
	EDP [nJ·s]	0.03	0.18	$\cdot^{-1}$	0.41	0.03	0.66
GPU fp16 (standard)	latency [μs]	5.5	5.6	7.9	19	40.5	59.5
	energy [μJ]	509.5	506	682	1697.6	3397.7	5095.3
	EDP [nJ·s]	2.8	2.83	5.38	32.25	137.62	303.2
GPU fp16 (optimal)	latency [μs]	6.9	6.9	7.7	21.5	35.3	56.8
	energy [μJ]	83.9	80.7	95.8	260.4	247	507.5
	EDP [nJ·s]	0.58	0.56	0.73	5.61	8.72	28.8

¹The long- and short-term components both affect the conductance of the same device so that the addition W+F does not need to be explicitly performed.

²The input-weight multiplication is computed via $I = G \cdot U$, which consumes power during the read operation. This power consumption is, however, already accounted for by the Decay term, which requires the application of a constant bias V_{bias} to control the Λ parameter. Here, the worst-case scenario is assumed ($V_{bias} = 0.6\text{V}$ applied to all synapses). The electrical current due to this voltage can be read out, giving the result of the input-weight multiplication. A crossbar array configuration is necessary to enable the full vector-matrix multiplication of all inputs with all synapses.

a)			b)		
Energy	Memristor	Memristor (limit)	EDP	Memristor	Memristor (limit)
GPU fp16 (standard)	966x	9200x	GPU fp16 (standard)	7.3x	456x
GPU fp16 (optimal)	96x	916x	GPU fp16 (optimal)	0.7x	43x

Fig. S19 Improvement factor of memristor over GPU in a) energy consumption and b) EDP for different implementations.

References

- [1] Heisig, T. *et al.* Oxygen Exchange Processes between Oxide Memristive Devices and Water Molecules. *Advanced Materials* **30**, 1–7 (2018).
- [2] De Souza, R. A. Oxygen Diffusion in SrTiO₃ and Related Perovskite Oxides. *Advanced Functional Materials* **25**, 6326–6342 (2015).

- [3] Zurhelle, A. F. Modeling the oxygen transport at heterointerfaces for oxide-based electronics. Tech. Rep. (2023).
- [4] Kreuer, K. D. Aspects of the formation and mobility of protonic charge carriers and the stability of perovskite-type oxides. *Solid State Ionics* **125**, 285–302 (1999).
- [5] Sata, N., Hiramoto, K., Ishigame, M. & Hosoya, S. Site identification of protons in: Mechanism for large protonic conduction **54**, 15795–15799 (1996).
- [6] Erickson, M. A., Maramara, L. A. & Lisman, J. A single brief burst induces GluR1-dependent associative short-term potentiation: A potential mechanism for short-term memory. *Journal of Cognitive Neuroscience* **22**, 2530–2540 (2010).
- [7] Moraitis, T., Sebastian, A. & Eleftheriou, E. Short-term synaptic plasticity optimally models continuous environments (2020). URL <http://arxiv.org/abs/2009.06808>.
- [8] Tsodyks, M. V. & Markram, H. The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability. *Proceedings of the National Academy of Sciences of the United States of America* **94**, 719–723 (1997).
- [9] Rodriguez, H. G., Guo, Q. & Moraitis, T. *Short-term plasticity neurons learning to learn and forget*, Vol. 162 of *Proceedings of Machine Learning Research*, 18704–18722 (2022). URL <https://proceedings.mlr.press/v162/rodriguez22b.html>.
- [10] Van der Wijngaart, R. & Oh, F. Boosting Application Performance with GPU Memory Prefetching. <https://developer.nvidia.com/blog/boosting-application-performance-with-gpu-memory-prefetching/> (2022). [Online; accessed 11-April-2024].