Neuromorphic Decoders: Paving the way towards adaptive, low-power and low-latency Brain Computer Interfaces.

by

Marco Rasetto

M.S., University of Genova, 2018

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2024

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Marco Rasetto

It was defended on

April 25th 2024

and approved by

Neeraj J. Gandhi, PhD, Professor, Department of Neuroscience

Rajkumar Kubendran, PhD, Professor, Department of Electrical and Computer Engineering

Zhi-Hong Mao, PhD, Professor, Department of Electrical and Computer Engineering

Ryad Benosman, PhD, Professor, Meta Reality Labs

Dissertation Director: Andrew Schwartz, PhD, Professor, Department of Neurobiology

Neuromorphic Decoders: Paving the way towards adaptive, low-power and low-latency Brain Computer Interfaces.

Marco Rasetto, PhD

University of Pittsburgh, 2024

Intracortical Brain-Computer Interfaces (iBCIs) intercept neuronal signals, allowing paralyzed individuals to perform movements and regain daily function. However, these advancements are mostly confined to laboratory settings due to high power consumption and bandwidth requirements for communication and computation in decoders, limiting their portability.

Decoders are often trained offline, requiring significant memory to store neural recordings, and are typically implemented on standard hardware architectures, which increases latency and power consumption. This research aims to develop mobile BCIs with low-power, low-latency decoders that can be used outside labs or hospitals.

Neuromorphic decoders present a promising solution by addressing power and latency constraints. These architectures process spiking data from iBCIs directly, eliminating the need for binning or spike counting and reducing latency and power consumption. Hierarchy of Time-Surfaces (HOTS) is particularly promising for BCI applications, using clustering to analyze data patterns, potentially making HOTS more interpretable than other machine-learning techniques. Online clustering may offer solutions to continual and incremental learning challenges, allowing BCIs to adapt to shifts in neural activity and new tasks without recalibrating or retraining decoders.

However, HOTS presents some challenges: It requires exponential decay kernels that are difficult to implement efficiently on digital hardware, and it shows lower accuracy compared to backpropagation-based spiking neural networks. Additionally, HOTS's current learning rule does not support continual and incremental learning.

This thesis addresses these issues. For hardware, it explores using electrochemical (ECRAM) memristor dynamics to implement exponential decay in HOTS decoders, potentially reducing circuit complexity and energy consumption. For software, it proposes

Sup3r, a learning algorithm that improves accuracy and skips uninformative events, enhancing efficiency and stability in online learning. Sup3r demonstrates continual and incremental learning, making it a fundamental advancement for HOTS models, applicable beyond BCI to various fields.

The hope is that these combined solutions will pave the way for low-power, adaptive neuromorphic decoders, enabling patients to regain autonomy outside the laboratory setting.

Table of Contents

List of Tables

List of Figures

Preface

I recall an IGN interview [43] with Gabe Newell, the founder of Steam, in which he was asked to comment on his experience developing Half-Life, one of the most successful and genre-defining games of our time.

Surprisingly, Gabe remembered his disappointment with the project because, as a developer, you don't have a full view of the product you're making. You mostly remember the missing features you couldn't include before the deadline.

Even though my Ph.D. is everything but "genre-defining", I can understand what Gabe meant in that Interview. This was a difficult project, born out of the necessity of writing a thesis on a topic that had to include the scientific work done before the proposal, fit the expectation for a biomedical engineering thesis, but also be general enough to keep me on the tracks of neuromorphic engineering. The last year was especially tough since my time had to be cut short by the absence of funding after my former advisor left the lab to reach for sunnier horizons. This problem convinced me to leave the US as soon as possible and focus on finding where to go next, further reducing the time I had left to work on the thesis.

For these reasons, I, too, see only the missing features.

Before writing this page, I asked myself many times if this Ph.D. was truly worth it, and while at first glance you might conclude that the answer is "no," the jury is still out.

This is because a Ph.D. is not only the papers we write or the time we spend with our advisors. In Pittsburgh, I grew as a person and built wonderful memories with my wife Chiara, who I want to thank first for always being my "pilastro centrale". Her love and support always fished me out of the darkest waters. The same could be said about my family: my mom, Daniela, my father, Valerio, and my sister Alice, aka Nana (which means small in Italian); you always made me feel close to you, even a quarter of the planet Earth away. In Pittsburgh, I have met some incredible people (a non-comprehensive list): Arianna, Erick, Lee, Luigi, Matteo, Philip, Daniel, Camila, Monique, Aaron (the best dungeon master I have ever met), Shantanu, Jiwon, Adiya, Anna (for the amazing Japanese recipes and for letting me believe my explanations made sense), Shruti, Becca Steph and August (with whom we

just had a wonderful trip in Sicily).

Last year's conferences also allowed me to build friendships across time zones with Jason Eshraghian, the green ogre of Neuromorphic engineering, and my once-lost-but-now-refund brother Hugo Ladret. I will never forget Maxime Fabre's mad cooking skills or the nurturing Willian, who saved me from the altitude-induced nausea of Telluride. I Love you all.

This Ph.D. would not have been possible without the plethora of amazing professors and scientists that I met at Pitt and CNBC. The first thank you goes to my advisor, Andy Schwartz; while I was struggling and feeling my passion wavering, I could always mend myself with a morning chat with you, where I felt I learned the most. Another colossal thank you goes to Himanshu Akolkar, the northern star I could always point at whenever I was lost at sea; you truly have the qualities of an excellent advisor. Another incredible thanks go to my committee, especially Neeraj Gandhi, for hearing my ramblings and allowing me to eject myself from a burning zeppelin (my former lab). I also want to thank some of the best professors I had the chance to meet at Pitt: Caroline Runyan, Aaron Batista, Helen Schwerdt, and Carl Olson. Thanks to you, I am a better scientist, and you will always be a source of inspiration for my work.

Finally, I want to thank my new partners in crime, Alessandro Milozzi and Michele Mastella. You reminded me to dream and shoot for the stars.

To you, the reader, a TLDR. If you are a Ph.D. student and you feel disappointed by your thesis, remember that you are not alone, that every lesson learned is not lost, and that a Ph.D. is so much more than a thesis.

Introduction

A brain-computer interface (BCI) is a system that allows direct communication between the brain and an external device, such as a computer or a prosthetic limb, bypassing the peripheral nervous system and conventional controls. BCIs are actively researched as a promising technology that might mitigate the effects of several life-altering pathologies, such as stroke and locked-in syndrome, restoring movements and sensation [86, 118, 123]. More-

over, BCIs are a fundamental neuroscientist's tool to understand the brain's inner workings and approach fundamental questions on brain physiology and pathology, such as understanding how the brain processes and encodes information and ultimately generates behavior [86, 118, 123]. Among BCIs, Intracortical brain-computer interfaces allow the highest spatial and temporal resolution, as they can record single neuronal action potential thanks to electrodes that are placed directly in the cortex, making iBCIs "the highest level of control for BCIs applications"[86]. For example, in 2021, iBCIs have been shown to reach up to 90 characters/min in decoding letter handwriting from the motor cortex with more than 90% accuracy [115], while ECoG BCIs, based on electrodes placed on the brain surface with a lower spatial resolution, have been shown to reach ≈47% on a limited 50 words vocabulary [63] in the same year. Thanks to their precision, iBCIs have been used to control robotic arms in 3D [87] achieving both reach and grasp on human patients [39, 18, 117]. iBCIs have also been used to restore patients' control over their limbs when combined with muscle stimulation [10, 2], and are currently being investigated to decode complex tasks like vocal articulation for speech decoding and synthesis [99, 98].

Even though iBCIs have been known for more than 40 years, their availability outside research laboratories remains limited [49, 123, 96]. Because of their low portability [79, 49, 123, 96], their use is often restricted to brief sessions in controlled environments. The same problem can be identified in animal research, where implanted animals need to be tethered to the recording equipment or in the proximity of wireless stations[96], making untethered free-moving brain decoding impossible.

This is because transmission of neural data from intracortical electrodes requires high bandwidth (≈46Mbit/s on a single 96-electrode device from Simeral et al. [96]) to accommodate the vast amount of information generated by neuronal activity. However, bandwidth is not the only problem associated with wireless transmission, especially when aiming at implantable devices that could one day be placed within the skull or under the scalp [64, 79, 124]. In these devices, power must be limited to keep the temperature increase of the nearby tissue below 0.5C [66]. One obvious solution is to reduce the need for energy-expensive communication by compressing the electrode signal after amplification and cleaning[113]. This approach could prove even more advantageous by moving the entire decoding stack on-device since

the information transfer rate once movement intention is decoded in the few bits per second when decoding 2d cursor position text speech and handwriting[79].

Unfortunately, this is still impractical due to the poor integration of neural decoders. While many decoders assume simple relations [11] between neural activity and decoded quantities, they train offline, which requires storing several neural recordings in memory. Moreover, training is often required multiple times a week because of the shift in neural activity[72, 22, 27]. Additionally, the neuron firing rate is estimated via time binning, which introduces latency and computational overhead. For these reasons, neural decoders have invariably been implemented on Von Neumann architectures that separate CPU, memory, and storage, reducing integration and increasing power consumption and latency.

Neuromorphic hardware represents one possible solution to integrate the decoding stack on-chip while reducing power consumption [92, 94, 12, 8, 25]. Neuromorphic devices are hardware-based neural networks that are inspired by computational models of spiking neurons. Similarly to biological neurons, neuromorphic neurons encode information through the analog timing between asynchronous digital pulses ("events" or "spikes"). This makes them able to directly interface with the action potentials from iBCIs, removing the need for time binning. Neuromorphic hardware is also highly energy efficient, with analog implementations in literature reaching as low as 10pJ per spike on the old 180 nm process [65].

Among different neuromorphic architectures, Hierarchy Of Time-Surfaces (HOTS) [51] might offer other interesting properties for iBCIs. HOTS networks are trained by clustering patterns of spikes (time-surfaces) and treating each neuron as a centroid of the clustering algorithm, similar to bag-of-words [19]. This feature makes HOTS a good candidate for neuromorphic decoders because clustering might offer better explainability than conventional SNNs trained with backpropagation. Moreover, clustering offers an intuitive implementation of continual and incremental learning[9], meaning that future decoders based on this algorithm might be able to re-calibrate during decoding and learn new tasks without forgetting previous ones.

However, HOTS architectures suffer from different fundamental problems. Time-surfaces are generated by interpolating spikes with exponential decay kernels. These kernels require digital clocked counters to calculate approximate linear decay or memory accesses to

store and retrieve spike timings before computing the exponential decay through specialized hardware blocks or look-up tables, making HOTS implementation on digital hardware unpractical and energy inefficient. Moreover, HOTS networks have a notorious gap with backpropagation-based Spiking Neural Networks in terms of accuracy and training time, as layers are trained sequentially before finally training an external classifier. This is because, similarly to bag-of-features models, HOTS networks need a standard classifier (like a support vector machine or a linear classifier) to classify input data[51, 81, 106], by accumulating the state of the network over a predefined time-bin. This makes HOTS not entirely compatible with neuromorphic hardware and makes other properties of clustering-based algorithms, such as continual and incremental learning, currently inaccessible.

In this thesis, I explored possible solutions to these problems in order to make HOTS a viable solution for the next generation of neuromorphic decoders.

In Chapter 1, I present a novel approach to implementing time-surface exponential decay using the volatile properties of an electrochemical memristive (ECRAM) device. Memristors are currently being researched as a way to implement weights for neural network accelerators [50, 104, 4, 108, 114] and to implement local learning rules such as clustering[48] and STDP [5]. Among these devices, ECRAM memristors are a promising solution to increase precision for stored weights while reducing power consumption for write and read operations. Combined with the findings in Chapter 1, ECRAM devices could offer an all-in-one solution to implement exponential decay in time surfaces, store HOTS centroids, and compute local learning rules, allowing for compact and trainable HOTS decoders.

Chapter 2 introduces Sup3r, a learning algorithm that enables HOTS networks to be trained end-to-end without the need for additional classifiers. Sup3r is designed to be local, so it does not require passing global data, such as weights and centroids, to other network centroids during training. Additionally, Sup3r is an online learning rule, which means it can be trained on a single sample of data at a time. These combined properties make it a viable solution for on-chip learning and capable of leveraging memristors, such as those used in Chapter 1, to train locally. Preliminary results show that Sup3r-HOTS reduces the performance gap against backpropagation while cutting uninformative events, potentially reducing unnecessary computation in HOTS networks. Additionally, I demonstrate that

Sup3r can grant HOTS networks the ability to adapt to data distribution shifts (continual learning) and learn new tasks without forgetting (incremental learning).

1.0 Using ECRAM Devices To Implement Temporal Dynamics in HOTS Networks

This chapter is the fusion of two papers I worked on during a collaboration with Feng Xiong's lab from the ECE department here at Pitt with the aim of developing a neuromorphic chip with a novel ECRAM device called LiWES (Lithium WO_3 Electrochemical Synapse). While the scope of the project shrank considerably, it demonstrated the feasibility of using the volatile properties of this class of devices for Spiking Neural Networks. In the first paper [109], I ideated and developed a toy problem to highlight the use of LiWES on a single neuron SNN (Here reported as section1.2.2). In the second paper [82], I simulated a full-sized HOTS network using LiWES volatile properties and tested its accuracy on widely used neuromorphic datasets while also analyzing the effect of device stochasticity and short-term dynamics on the proposed task. The remaining sections of this chapter are from this second paper.

1.1 Introduction

The last decade has brought considerable progress in AI, mainly owing to the advent of Graphics Processing Units (GPUs) and other hardware accelerators. However, this progress has not been matched from the perspective of emulating general intelligence and cognition. Ideas such as deep multilayer learning and backpropagation have helped solve a particular class of well-defined problems but require high energy and vast amounts of labeled data [74, 40]. These requirements drastically limit on-board "intelligence" and reduce autonomy. Thus, essential functionalities such as continuous always-on learning with a reasonable power budget are still out of reach.

Neuromorphic engineering holds the promise of mitigating these restrictions [85, 58]. Recently, this field has reached a level of maturity that allows it to impact several other domains where autonomy and low power on-the-edge computation are crucial. One core

principle is to remove the separation of memory and computation, typical of Von Neumann architectures, by taking inspiration from neurons and synapses and more closely integrating computation and memory.

Many researchers have taken an interest in memristive devices, given their ability to implement tunable nonvolatile weights similar to synaptic efficacy in biological synapses. Following this paradigm, memristors have been applied to many network-based computational approaches. Doygu et al. simulated memristor networks able to learn sequences of inputs [50]. Suri et al. used another simulated network of phase-changing memristors to learn MNIST letters[104]. Systems that perform STDP (Spike Time Dependent Plasticity) using RRAM (REsistive RAM) devices [4] or that implement recursive networks using PCM memristors have also been reported [26]. Vincent et al. simulated a network of STT-MRAM devices for car detection [108]. Networks of memristors have also been proposed to implement the k-means algorithm [48] and unsupervised learning [114]. In all of these examples, the synaptic devices modeled using memristors working in a "static" fashion, i.e., as a fixed scalar multiplier of spike events, before integration by the "neuron membrane". This approach has been widely adopted because of its simplicity of implementation and mathematical tractability, as synaptic operations can be described by simple linear algebra.

In recent years, a different approach has surfaced. Several works have demonstrated the presence of transient conductance responses in memristive devices akin to short-term plasticity (STP) and Excitatory/Inhibitory Post Synaptic Potentials (EPSP/IPSP)[119, 111, 121, 67]. These memristors with "volatile" properties are extremely promising for implementing neuromorphic networks that need physical devices capable of temporal computation [7]. In these devices, input events (i.e., voltage pulses) cause temporary changes in conductance that exponentially relax back to baseline, like EPSPs and IPSPs in biological synapses. Moreover, the conductance change is potentiated when multiple input events are close in time (STP). These short-term dynamics allow the modeling of temporal kernels and short-term plasticity without the need for additional circuitry. Electrochemical memristors (ECRAM) exhibit both STP and EPSPs simultaneously [93, 125]. However, in memristors with oscillatory properties, [41, 59, 73], STP is not present. Moreover, some devices can produce EPSPs with multiple exponential decays, making the range of possible dynamics extremely complex.

We are still far from understanding the entire repertoire of short-term dynamics in volatile memristors, let alone being able to exploit them in real, practical scenarios. Many works utilizing the short-term dynamics of memristive devices have demonstrated only simple networks (or even single neurons) operating on limited examples[103, 83, 110, 102, 7, 121, 67], or use memristor dynamics for different spiking neuron operations such as adaptive thresholds [91] and synaptic traces [23]. At the time of writing, the relationship between the different types of dynamics in representing temporal information from spiking data and recognition rates on real-world neuromorphic datasets has not been studied. Understanding this relationship will enable the design of custom neuromorphic systems that make full use of memristive dynamics for efficient computation.

This chapter focuses on an ECRAM device called LiWES (Lithium WO_3 Electrochemical Synapse), which demonstrated different types of volatile dynamics (STP and multiple exponential decays) in a range of ten to hundred milliseconds (required by many neuromorphic datasets), high precision of 1024 states, and low reading power [110].

Section 1.2 briefly introduces this device and its properties. Section 1.2.1 presents a mathematical model to simulate the device response, which is then used to test the device behavior on a small toy problem in section 1.2.2. In the following sections, the same model is also used to simulate a full Hierarchy of time-surfaces (HOTS) architecture [51] on more complex datasets (N-MNIST [68] and POKERDVS [89]). These sections focus on answering the following questions:

1) Can we use LiWES pulse response to implement HOTS exponential decay operations?

2) Is the device stochasticity reducing the network accuracy?

3) Are STP and multiple exponential decays useful properties for SNN architectures?

1.2 Testing LiWES, A Li_xWO_3 Electrochemical Synapse For Spiking Neural Networks

"Volatile" memristors enable efficient implementations of temporal computing, as they combine temporal dynamics and short-term plasticity in a single device. Among these, electrochemical memristors [29] have become good candidates thanks to their low power consumption, linear and symmetric response, low variability, and high reliability [29, 55, 30, 120, 125, 77, 112, 107, 122, 57, 56]. Wan et al. previously proposed a novel electrochemical memristor [110] based on Lithium Ions and Tungsten Oxide (Li_xWO_3). This memristor has the advantages of low programming voltage (0.2 V), fast programming speed (500 ns), and high precision (1024 states corresponding to 1024 10ms 0.5V "write" pulses before reaching saturation), wide conductance range ($\sim 1\mu$s to $\sim 200\mu$s), with a channel area of 400x200μm^2.

These devices have been used to model synapses and to implement electrochemical random access memory (ECRAM) [105]. They are especially suitable for neuromorphic networks because they can model synaptic dynamics and short-term plasticity (STP) with time constants ranging from a few to hundreds of milliseconds. We focus on a version of Li_xWO_3 memristor that uses a self-gate design in which transitory effects dominate long-term effects[112].

This structure of the Li_xWO_3 memristor is illustrated in Fig. 1(a). Unlike conventional two-terminal memristors, the Li_xWO_3 electrochemical is composed of three terminals, the (S)ource, (D)rain and (G)ate. The memristor is built by deposition of tungsten oxide (WO_3) films on a $LaALO_3$ substrate. Lithium ions introduced via an electrolyte gel can flow between the gate and channel. When embedded into the tungsten oxide films, they act as short or long-term doping charges, changing the film conductance [110]. The conductance between the source and the drain terminal, G_{DS}, is considered to be the synaptic weight of the device.

The memristor's electrical behavior is illustrated in Fig. 1(b). The conductance G_{DS} can be read by applying a small DC reading voltage ($V_{\mathrm{DS}} = 100\mathrm{mV}$) across the source and drain. During the "write" phase, we modulate G_{DS} by applying a square voltage pulse between the gate and drain. The channel conductance increases linearly. Once the pulse is removed, the conductance decays following a double exponential decay function. Reading can be carried

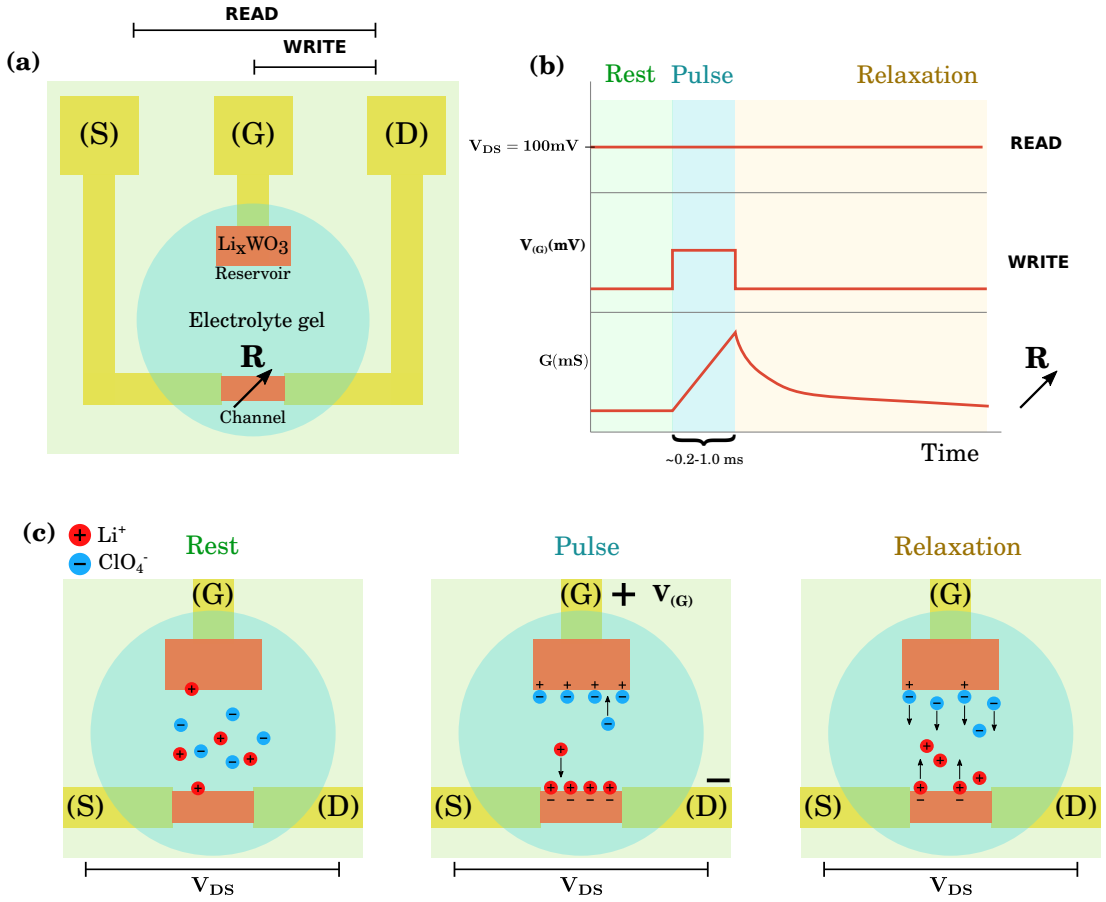

Figure 1: The structure and operation of the physical Li_xWO_3 electrochemical synapse modelled in this paper. (a) The Li_xWO_3 electrochemical synapse is a three-terminal device with a (S)ource, (G)ate and (D)rain. The gate and channel between S and D are built by deposition of tungsten oxide (WO_3) films (red) on a $LaALO_3$ substrate (green). The films are connected to gold terminals (yellow). Lithium (Li+) and ClO_4- ions are introduced by applying a drop of electrolyte gel (blue) on top of the device. (b) The electrical behavior of the memristor in response to a square WRITE voltage pulse applied between (G) and (D) and a small DC READ voltage (0.1 V) applied between (S) and (D). (c) The electrochemical behavior of the memristor.

out in parallel and does not interfere with writing. This design allows for lower programming voltages and better state retention [107]. State retention and low programming voltage are important when designing standard memristive networks for spiking and artificial neural network implementations. These allow for stable networks with low power consumption. Multiple small pulses can be used to change the device conductance gradually, allowing the artificial networks to mimic their biological counterparts.

Fig. 1(c) illustrates the electrochemical operation. At rest (no gate to drain voltage applied), the Li+ and ClO_4- ions are in equilibrium between the tungsten oxide films and the electrolyte gel. The positive "write" pulse creates an electric field, which causes Li+ ions to accumulate at the channel/electrolyte interface and charge-balancing ClO_4- ions to accumulate at the gate. Doping of the channel by the Li+ ions increases the channel conductance. During the relaxation phase, after the removal of the write pulse, the ions return to equilibrium and the channel conductance returns to its resting value. The double exponential decay response can be explained by the Kohlrausch-Williams-Watts (KWW) relaxation model [116, 53], which has also been found in other electrochemical devices [125].

Changing the material properties enables "programming" the exponential decays in the range of tens to hundreds of milliseconds, which is optimal for the temporal integration of events for many real-world datasets [51, 76]. Due to the double exponential decay, closely spaced pulses generate accumulation. This property allows the memristor to exhibit short-term plasticity (STP) similar to biological neurons. Moreover, the high number of pulses required to reach saturation (1024) makes it more than capable of working on the proposed datasets, where the maximum number of events per pixel is 48, corresponding to a max of 48 "write" pulses per single synapse/device. Together, these properties make this device an excellent candidate for studying the computational benefits of memristors' dynamics for neuromorphic time-based learning applications.

1.2.1 Mathematical Model Of Synaptic Dynamics

Following [116, 53, 112, 110, 125], we developed a mathematical model to predict the electrical behavior of the Li_xWO_3 memristor by first modelling its response to a single square

write pulse, then combining the responses.

Given a spike train with spike times $\{t_i\}$ indexed by $i \in \{0, 1, 2, \ldots\}$, the conductance response is given by

$$G(t) = \sum_i \left(G_{1,i}(t) + G_{2,i}(t)\right) + \eta(t) \tag{1-1}$$

where $G_{1,i}(t) + G_{2,i}(t)$ is the memristor's response to all spikes up to and including the ith spike for times $t \in (t_i, t_{i+1}]$, and $\eta(t)$ is zero-mean Gaussian white noise with variance σ^2. To model the double exponential decay, we express the response as the sum of two components, $G_{k,i}(t)$ for $k \in \{1, 2\}$, each modeling a single exponential response.

We define the components $G_{k,i}(t)$ recursively:

$$G_{k,i}(t) = \begin{cases} L_{k,i}(t) & \text{for } t_i < t \leq \min(t_i + w, t_{i+1}) \\ E_{k,i}(t) & \text{for } t_i + w < t \leq t_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{1-2}$$

where $w > 0$ is the width of the write pulse. $L_{k,i}(t)$ models the linear rise in conductance starting from $G_{k,i-1}(t_i)$ to $G_{k,i-1}(t_i) + A_{k,i}$, where $A_{k,i}$ is the peak conductance change due to the ith write pulse.

$$L_{k,i}(t) = G_{k,i-1}(t_i) + A_{k,i}\left(\frac{t - t_i}{w}\right) \tag{1-3}$$

$E_{k,i}(t)$ models an exponential decay in conductance from $G_{k,i-1}(t_i) + A_{k,i}$ to zero with time constant $\tau_{k,i}$.

$$E_{k,i}(t) = \left(G_{k,i-1}(t_i) + A_{k,i}\right) e^{-\left(\frac{t - t_i - w}{\tau_{k,i}}\right)} \tag{1-4}$$

Fig. 2 shows our model for a given pulse width w at time t_i.

We can see from (1-2), (1-3), and (1-4) that $G_{k,i}(t)$ depends on the conductance at the start of the ith pulse, $G_{k,i-1}(t_i)$. This models STP, where past pulses all contribute to the current device conductance. To model the hypothetical memristor without STP in section IV C, we remove $G_{k,i}(t)$ from (1-3) and (1-4)

$$L_{k,i}^{\overline{\text{stp}}}(t) = A_{k,i}\left(\frac{t - t_i}{w}\right) \tag{1-5}$$

$$E_{k,i}^{\overline{\text{stp}}}(t) = A_{k,i} e^{-\left(\frac{t-t_i-w}{\tau_{k,i}}\right)} \tag{1-6}$$

so that every new pulse resets the peak conductance to $A_{k,i}$, rewriting any conductance value from previous pulses.

As we are also interested in studying the effect of device stochasticity on computation, we consider two models: an ideal model and a stochastic model.

In the ideal model, the peak conductance changes and time constants are the same for all pulses, i.e., $A_{k,i} = \overline{A_k}$ and $\tau_{k,i} = \overline{\tau_k}$ for $k \in \{1, 2\}$, where $\overline{A_k}$ and $\overline{\tau_k}$ are positive constants. The noise is zero ($\eta(t) = 0$).

In the stochastic model, the $A_{k,i}$ and $\tau_{k,i}$ are drawn from an independent and identically distributed discrete time (i.i.d.) random process in i, where each sample is drawn from a Gaussian distribution,

$$A_{k,i} \sim \mathcal{N}(\overline{A_k}, \sigma_{A_k}^2). \tag{1-7}$$

$$\tau_{k,i} \sim \mathcal{N}(\overline{\tau_k}, \sigma_{\tau_k}^2). \tag{1-8}$$

where negative samples are rectified.

The dynamics of the LiWES can be tuned by changing the write pulse properties, such as pulse width and amplitude [110]. To replicate the device behavior for a given pulse, we need to obtain the mean and standard deviation of the model parameters $(A_1, A_2, \tau_1, \tau_2)$. We first compute the rest conductance by averaging the conductance before pulse onset and subtracting this from the data. We then combine (1-1) and (1-2) to compute the response, which we split into two parts:

$$G_{\text{rise}}(t) = (A_1 + A_2)\left(\frac{t - t_0}{w}\right) \text{ for } t_0 < t \leq t_0 + w \tag{1-9}$$

$$G_{\text{decay}}(t) = A_1 e^{-\left(\frac{t-t_0-w}{\tau_1}\right)} + A_2 e^{-\left(\frac{t-t_0-w}{\tau_2}\right)} \text{ for } t_0 + w < t \tag{1-10}$$

We fit the model parameters $(A_1, A_2, \tau_1, \tau_2)$ using the least squares fit between the experimental data after the write pulse ($t > t_0 + w$) and $G_{decay}(t)$ using the Gauss-Newton algorithm. We use the model and fitted parameters to predict the responses for $t \leq t_0 + w$.

To find the parameters of the stochastic model, we repeated the model fit using data from multiple recordings of pulses with different pulse amplitudes (ranging from $1V$ to $4V$) and

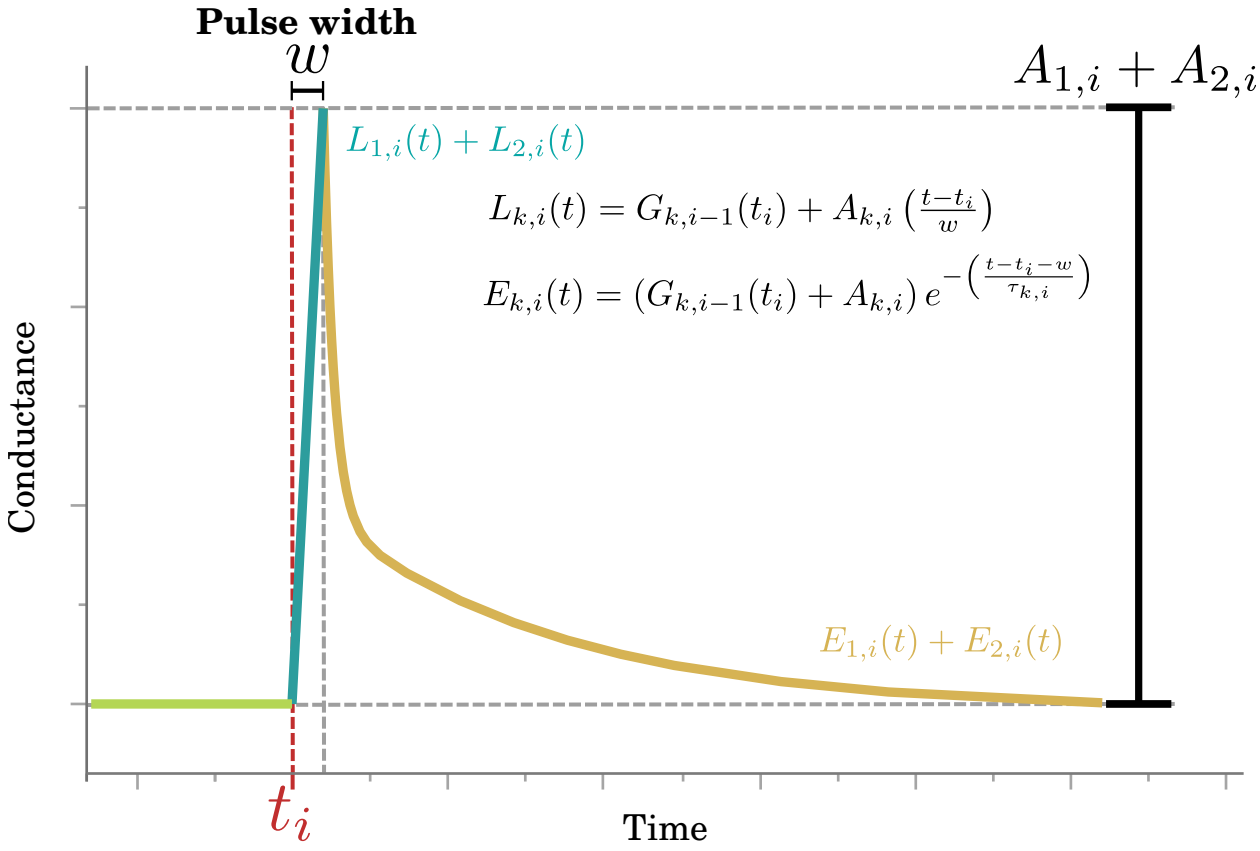

Figure 2: Example response of our model to a pulse at t_i with pulse width w assuming initial steady state (in green). The model presents a linear increase (in blue) for the duration of the pulse. At its end, it will relax to a steady state following a double exponential decay (in gold).

Table 1: Model parameters (mean ± s.d.) obtained with a pulse with fixed 1V amplitude but varying duration.

1V	A_1	τ_1	A_2	τ_2	σ_η
200us	0.57±.27	5ms±2ms	0.5±.05	92ms±18ms	0.0464
500us	0.74±.18	16ms±9ms	0.26±.06	588ms±31ms	0.0245
750us	0.78±.19	10ms±2ms	0.23±.03	513ms±98ms	0.0149
1ms	0.75±.29	10ms±3ms	0.22±.02	390ms±68ms	0.0097

durations (ranging from 200µs to 1 ms). Based on these fits, we calculated the distributions of the model parameters. We report the fitting results in Table 1 and Table 2. The tables show the mean and standard deviations of the parameter estimates fitted over 20 recordings for the 200us 1V pulse and 5 recordings for the remaining conditions. The variance of the noise σ_η was set to the mean square error of the fit. For details, refer to the supplementary materials.

We can use equations (1-1)-(1-8) and the distributions specified in Tables 1 and 2 to simulate the response of any number of devices to any set of spike trains. Figure 3 compares the output of our stochastic model with the response of an actual device to the same spike train. Note that due to the stochasticity, we do not expect spike-to-spike matching of the

Table 2: Model parameters (mean ± s.d.) obtained with a pulse with fixed 200us duration but varying amplitude.

200us	A_1	τ_1	A_2	τ_2	σ_η
1V	0.57±.27	5ms±2ms	0.5±.05	92ms±18ms	0.0464
2V	0.54±.29	7ms±1ms	0.35±.02	122ms±20ms	0.0244
3V	0.77±.24	13ms±6ms	0.23±.02	373ms±98ms	0.0189
4V	0.75±.17	11ms±3ms	0.25±.01	501ms±101ms	0.0159

responses. Rather, the statistics and timing of the responses will be similar.

1.2.2 LiWES Volatile Response For Temporal Computation

The goal of this section is to show how LiWES devices' dynamic behaviors could be used to boost classification performance in highly time-dependent scenarios. The principle behind the proposed computation is that when the LiWES devices receive a set of spikes, their conductance value will change depending on the temporal structure (individual spike timings) of the input spike train. Furthermore, in the absence of LTP when using Li_xWO_3 self-gate and channel, the conductance of the device will be uniquely determined by the input spiking pattern and the time of integration,[51, 3] granting the device the ability to integrate temporal information and distinguish between different spike patterns.

In standard neuromorphic SNNs with NO-STP synapses, the synaptic efficacy (or weight), which remains fixed during inference, is used to simply scale current pulses directed toward the post-synaptic neuron. In these models, the temporal integration of stimuli is left solely to the neuron, whereas in STP-enabled networks, synapses also encode temporal information through weight changes, enriching network dynamics[6, 61, 54] and increasing the ability of neurons to discriminate between temporal stimuli [13]. For this reason, when compared with NO-STP synapses, a network including the proposed LiWES device should increase its performance in highly time-dependent tasks, such as the classification of different spike patterns. To test this hypothesis, we propose a test tailored to compare LiWES to an IDEAL synapse (a noiseless LiWES device) and a standard NO-STP synapse. Here, we connect a post-synaptic neuron, modeled with Leaky Integrate and Fire profile (parametrized with the membrane decay constant τ_{m} and spiking threshold $= \infty$), to a pre-synaptic neuron, which is a Poisson Spike generator (Figure 4a). As shown in Figure 3, the channel conductance response of LiWES shows a spike profile, where the conductance quickly reaches the maximum conductance level followed by an exponential decay back to initial conductance level, due to ionic-gating governed STP effect. Thus, we are able to model the conductance response of LiWES as shown in Figure 4b.

In the proposed task, we generate multiple pre-synaptic neuron spike trains with a fixed

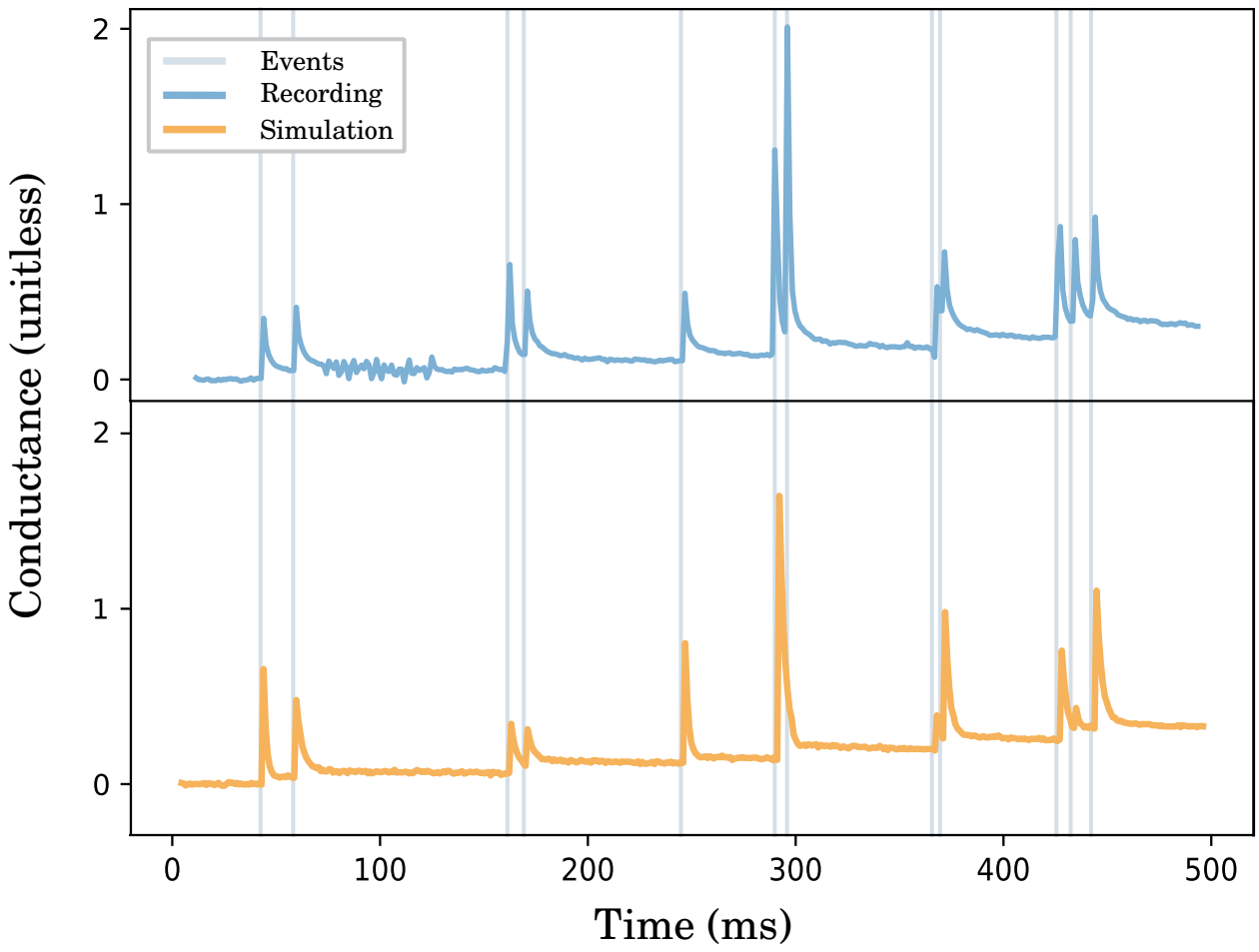

Figure 3: Our stochastic model (in orange) and a memristor recording (in blue) for a pulse of 1V 200us. We sample the response parameters from the parameter distributions obtained from fitting our model to experimental data.

maximum duration. As each spike sequence is randomly generated at a fixed frequency, therefore, it differs from the others mainly by its temporal characteristics (the timestamps of individual spikes), and it represents a single class of a classification problem. The beginning of each spike train is delimited by t_{onset}. A "sequence end spike" is added at the end of each spike train at a specific time t_{end} (Figure 4c), and the post-synaptic neuron membrane potential is read out at t_{read} (Figure 4d, representing the output of the system. Each spike train is presented to the synapse multiple times to obtain multiple membrane potential readouts for the same "class" (or spike pattern). To calculate the class separability of the readouts, we define a distance metric as the difference between the Euclidean distance of points between different classes (interclass distance) and the distance of the points within the same class (intraclass distance) in Figure 4e. As the membrane potential of the post-synaptic neuron is always read out at the same time (t_{read}) after the last spike (t_{end}), a neuron unable to integrate temporal information will have similar membrane potential for different spike patterns, and therefore, it will have an average interclass distance of zero or close to zero. However, for an STP-enabled neuron, its membrane value depends on previous spiking activity, which gives different values of interclass distance based on different classes. This is the case shown in Figure 4f, where a fast spiking neuron (τ_{m} = 10 ms) is stimulated with Poisson generated spikes at a slow mean of 10 Hz frequency. The number of classes used for this simulation was 50, each one presented ten times (for intraclass measurement), for a total number of 500 points. In this case, class separability (interclass distance–intraclass distance) is $\approx 3.8 \times 10^{\tau 4}$ for the NO-STP synapse, $\approx 4.9 \times 10^{-2}$ for LiWES device ($\approx$128$\times$ higher relative to NO-STP synapse), and $\approx 8.6 \times 10^{-2}$ for the IDEAL synapse ($\approx$226$\times$ higher compared with NO-STP synapse), with using a synaptic weight k of 4.3 (see the Experimental Section for model build details). As both comparison synapses (NO-STP and IDEAL synapses) are totally deterministic, their mean intraclass distance is 0. The same simulation parameters are used in Figure 4g for a much slower post-synaptic neuron (τ_{m} = 100 ms). Even though the post-synaptic neuron is relatively slower to integrate temporal information, a boost in class separation ($\approx$1.4$\times$ in LiWES and $\approx$1.7$\times$ in the IDEAL synapse, relative to the NO-STP synapse) can still be achieved owing to the natural stochastic STP in LiWES. The class separability is $\approx 8.4 \times 10^{-2}$, $\approx 1.2 \times 10^{-1}$, and $\approx 1.4 \times 10^{-1}$, for the NO-STP synapse,

LiWES device, and the IDEAL synapse, respectively, with using a synaptic weight k of 16.7. By implementing the temporal spiking information in STP of LiWES, we improve the pattern classification performance (up to 128× compared with NO-STP synapse) in highly time-dependent scenarios.

1.3 Memristor Implementation Of HOTS

The Hierarchy of Event-Based Time-Surfaces (HOTS) architecture is a neuromorphic architecture for unsupervised pattern recognition [51]. HOTS networks are highly versatile and can be applied to the output of neuromorphic sensors for different modalities [51, 42, 32, 81]. Moreover, HOTS neurons are more mathematically tractable than other Spiking Neural Network (SNN) models. This feature makes it easier to isolate the effects of memristive dynamics on network behavior using the analysis we describe in the next section.

A HOTS network maps each input spike to an output spike from one of the neurons in the network. HOTS networks draw inspiration from clustering. Each neuron in a HOTS network represents a cluster corresponding to a pattern of events within a spatial window. The diversity of possible input patterns is reflected by the diversity of patterns represented by the neurons in the network. In order to represent event patterns as points to cluster, HOTS introduces the concept of the Time-Surface. Every time a neuron produces an event, it triggers the creation of a Time-Surface, an array representing the time history of events at that and neighboring neurons.

This section describes a model of a HOTS implementation that exploits the dynamics of Li_xWO_3 memristors to create the time-surfaces. The process of creating time-surfaces is shown in Fig. 5, where we assume input comes from an event-based vision sensor and the task is digit recognition. Input images (a) are captured by the event-based sensor (b), which emit trains of events at each pixel in response to brightness changes (c)[51].

Each event i from an event-based vision sensor can be described by the tuple:

$$ev_i = (x_i, y_i, p_i, t_i) \tag{1-11}$$

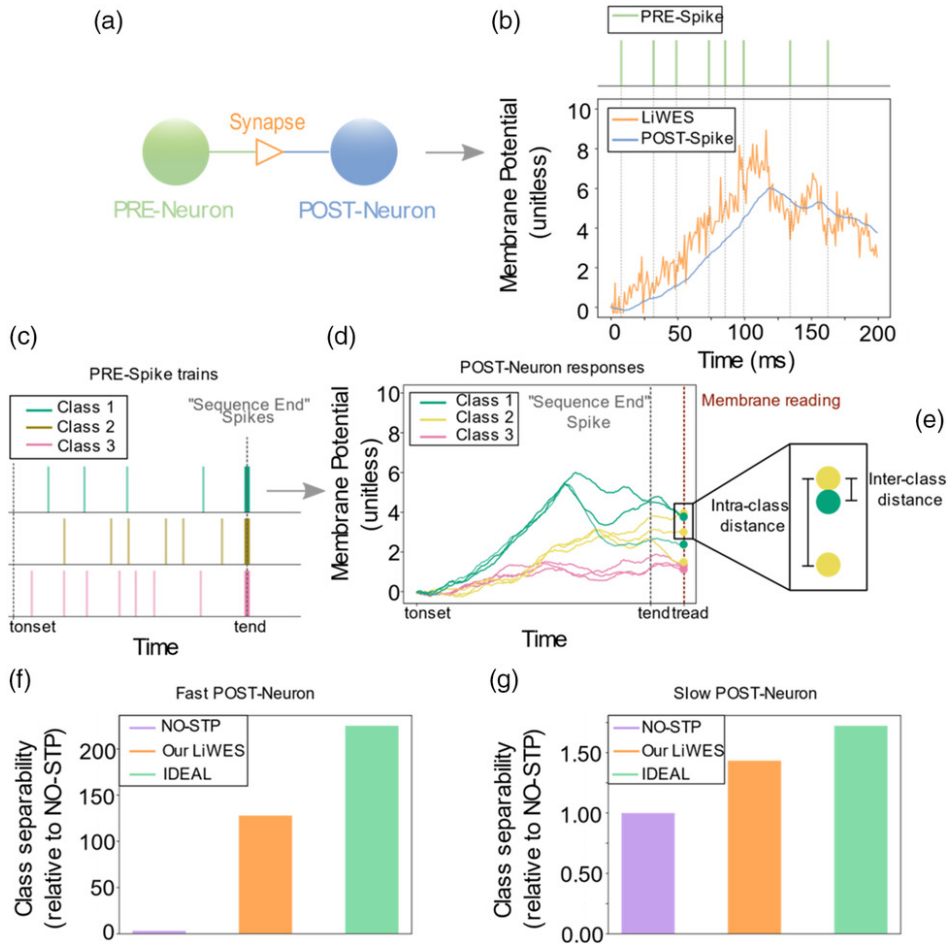

Figure 4: SNNs computation-based STP of LiWES. a) The diagram of network, a Poisson pre-synaptic (PRE) neuron connected to a Leaky and Integrate and Fire post-synaptic (POST) neuron through a synapse (NO-STP, LiWES or IDEAL synapse). b) An example of a Poisson train spike eliciting activity in LiWES and the consequently generated membrane potential. c–e) An example of the proposed spike-based SNNs computation model for classification performance benchmark. c) The PRE-Neuron produces multiple random spike trains; at the end of each one, we add a "sequence end" spike occurring always at the same timestamp (t_{end}). Each spike train represents a different class in a classification problem. d) We then record multiple POST-Neuron responses (three responses per each spike train), to better characterize the device noise and cycle-to-cycle variation, and finally, we save the membrane value after the "sequence end spike" (at t_{read}). e) Finally, for each point, we calculate the interclass distance between points of different spike trains and the intraclass distance between points of the same spike train class. These measures indicate how much each point position encodes for temporal information, and how well the points are separable in a classification task. f,g) The classification result of the benchmarked synapses. f) The classification comparison for a 10 Hz Poisson PRE-Neuron and a fast POST-Neuron ($\tau_{\text{m}} = 10$ ms). g) The classification comparison for the same 10 Hz Poisson PRE-Neuron but a much slower POST-Neuron ($\tau_{\text{m}} = 100$ ms).

where x_i and y_i are the pixel positions, p_i is the polarity (the direction of brightness change in the pixel), and t_i is the timestamp.

In the original version of HOTS, every incoming event gives rise to an instantaneous rise exponential decay kernel with no memory. In our model, incoming events give rise to double exponential decays with STP. We assign a LiWES memristor to each spatial location and polarity (x, y, p), and using the events at (x, y, p) to generate WRITE signals to that memristor, which generate changes in its conductance $G_{x,y,p}(t)$ following the model described in Section 1.2.1. For each event ev_i, we create a time-surface by sampling the conductance at all memristors within a square spatial window of lateral size l around (x_i, y_i), i.e.

$$S_i(m, n, p) = G_{x_i+m, y_i+n, p}(t_i) \tag{1-12}$$

for $m, n \in \{-(l-1)/2, \ldots, (l-1)/2\}$ and for all polarities p, where the subscripts indicate the memristor's location and polarity.

We use an unsupervised clustering method, such as the K-means algorithm, to cluster the time-surfaces. The clusters capture recurring spatio-temporal features of the input data. Each input event generates an output event at the same location and time, but whose polarity is given by the closest cluster. Thus, the number of possible polarities of output events is equal to the number of clusters.

We can define a multiple layer architecture by defining each layer k as a single iteration of this process. Its input events are

$$ev_i^k = (x_i^k, y_i^k, p_i^k, t_i^k) \tag{1-13}$$

Its output events are:

$$ev_i^{k+1} = (x_i^{k+1}, y_i^{k+1}, p_i^{k+1}, t_i^{k+1}) \tag{1-14}$$

where superscripts index the layer number, $x_i^{k+1} = x_i^k$, $y_i^{k+1} = y_i^k$ and $t_i^{k+1} = t_i^k$. The output events of one layer become the input to the next. However, to increase spatial integration from layer to layer, we often sub-sample the output events of one layer before inputting them to the next layer.

Fig. 5(g)-(m) shows an example of a two-layer architecture. In (g), we plot the most recent input events before a reference event ev_i^1. We sample the memristor conductances in

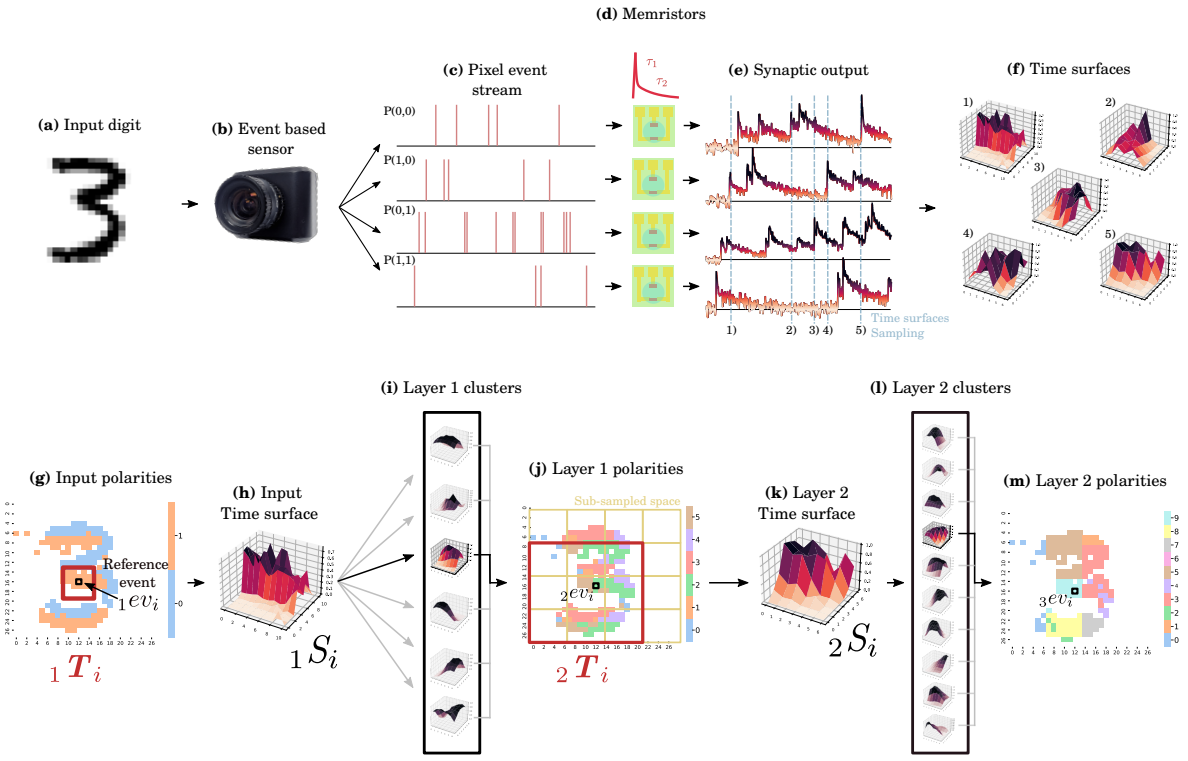

Figure 5: Top row (a-f): Mapping spatio-temporal event patterns into Time-Surface features using memristive synapses. Bottom row (g-m): N-MNIST classification using the memristive HOTS network. An input digit (a) is presented to the event-based sensor (b), which produces asynchronous event streams (c) at each pixel based on the luminance variation over time. Each pixel stream is input to a memristor (d), which interpolates the spike train with exponential decay kernels (in red) with time constants τ_1 and τ_2, resulting in time-varying conductances (e). For each new event ev_i (indicated in light blue), we generate a Time-Surface by sampling the conductances from memristors at neighboring pixels, resulting in a 2D map that encodes temporal correlations between events at different pixels (f). The spiking activity (g) from the event-based sensor has two polarities, indicating increases (orange) or decreases (blue) in luminance. This figure shows the polarity of the last spike at each pixel, if any, during the last 10ms before a reference event ev_i^1. time-surfaces (h) are created by sampling changes to memristor conductances in a square neighborhood around the reference event, shown in red. time-surfaces are clustered (i). Input events produce output events (j) with the same location and time stamp, but polarity determined by the closest cluster index. This new set of events is spatially sub-sampled, resulting in a larger effective neighborhood size. In the next layer, these are used as inputs to generate new time-surfaces (k) that are again mapped to clusters (l) to produce output events (m). This process can be repeated multiple times to increase the temporal and spatial scale of the classified features.

the square neighborhood around (x_i^1, y_i^1) (shown as the red square) to produce a Time-Surface S_i^1 (h, only one polarity shown). This Time-Surface gets assigned to a cluster of Layer 1 (i), producing a new output event ev_i^2 with a new polarity p_i^2 (j). Due to sub-sampling, time-surfaces in layer 2 usually correspond to larger effective neighborhoods in layer 1. The entire process can be repeated, as shown in Fig. 5(k,l,m), until we achieve a desired amount of temporal and spatial integration.

Similarly to [51], we create a feature vector for each spike activity recording by building a histogram $\mathcal{H}$ of the polarities of spikes from the last layer collected across all pixels and over the entire recording.

We classify the feature vector using a polynomial Support Vector Classifier (SVC). In our comparative experiments seeking to elucidating the effect of different facets of the memristor dynamics on the computed features, we used the simpler Euclidean distance approach proposed in the original HOTS paper [51]. For each label, we computed a template $\overline{\mathcal{H}}_{label}$ by averaging over all the histograms with that label in the training set. To classify new digits, we performed template matching using the Euclidean distance measure.

1.4 Experimental Results

1.4.1 The Effects Of Programmable Integration Constants On Accuracy

Tables 1 and 2 show that different pulse settings (Voltage and duration) give rise to different decay time constants. The ability to tune integration constants is fundamental for neuromorphic applications, as spike rates vary between different applications. While these results do not enable us to model the full relationship between pulse settings and time constants, they do enable us to investigate whether different applications benefit from different time constants.

We tested the two-layer network shown in Fig. 5(g to m) on the N-MNIST [68] and POKERDVS [89] datasets. The sub-sampling factor was 7, which reduces the N-MNIST resolution from 28x28 in layer 1 to 4x4 in layer 2 and the POKERDVS resolution from

Table 3: percent classification accuracy for different pulse parameters

Pulse settings	(200µs,1V)	(200µs,2V)	(200µs,3V)	(200ms,4V)	(500µs,1V)	(750µs,1V)	(1ms,1V)
N-MNIST	90.90 ± 0.22	90.92 ± 0.24	91.15 ± 0.37	90.93 ± 0.21	90.09 ± 0.16	90.95 ± 0.19	$\mathbf{91.27} \pm 0.29$
POKERDVS	$\mathbf{98.0} \pm 3.31$	96.50 ± 3.90	97.50 ± 4.03	$\mathbf{98.00} \pm 3.32$	97.00 ± 3.32	96.50 ± 3.20	97.50 ± 4.03

35x35 in layer 1 to 5x5 in layer 2. The Time-Surface lateral dimensions for N-MNIST results are $l^{[1]} = 7$ and, $l^{[2]} = 3$ respectively, for the first and second layer. The Time-Surface lateral dimensions for POKERDVS results are $l^{[1]} = 5$ and $l^{[2]} = 7$. The number of clusters for the N-MNIST results is $N^{[1]} = 32$ and $N^{[2]} = 96$. The POKERDVS network was smaller with only $N^{[1]} = 8$ and $N^{[2]} = 64$ clusters. To eliminate the effect of device stochasticity and enable comparison with other work on these datasets, which typically do not include device stochasticity, we used the ideal model described in Section 1.2.1.

Since our implementation of HOTS uses K-means for learning the time-surfaces, which requires relatively little data to train, we only use 10% of the training set for the N-MNIST results. Files were randomly selected at each run. However, our testing results are reported based on performance on the *entire* test set.

Table 3 compares the test-set classification accuracies on the two datasets for all the pulse settings listed in Tables 1 and 2. These results were calculated over 5 runs on N-MNIST and 10 runs on POKERDVS. We classified with a polynomial support vector machine of order 3. We report results with additional classifiers in our supplementary materials.

Our results show that we obtained the best performance on the N-MNIST and POKERDVS datasets using different pulse parameters, which resulted in very different time constants. The best performance for N-MNIST were obtained for 1V 1ms-long pulses, which gave time constants $\tau_1 = 10$ ms and $\tau_2 = 390ms$. The best performance for POKERDVS were obtained for 200µs-long pulses with amplitude either 1V ($\tau_1 = 5$ms, $\tau_2 = 92ms$) or 4V ($\tau_1 = 11$ms, $\tau_2 = 501ms$). These differences highlight the importance of the ability to tune time constants.

Our LiWES memristive HOTS network achieves state-of-the-art performance on N-MIST (91.27%), exceeding that reported by Sironi et al. [97] and Iyer et al. [46], despite the use of only 10% of the training data.

1.4.2 The Effects Of Stochastic Dynamics On Accuracy

To analyze the effects of stochastic dynamics on recognition rate on a neuromorphic architecture, we compared the performance of HOTS architectures on the N-MNIST dataset using the stochastic memristor model (the 'Noisy' network) and the ideal memristor model (the 'Ideal' network) defined in Section 1.2.1.

Since the implementation of in-situ learning on the memristive chip was beyond the scope of this paper, we limited noise analysis to inference only. Calculation and clustering of time-surfaces was performed using the Ideal network model only. The Noisy network and the Ideal network share the same sets of clusters, but in the case of the Noisy network, the Time-Surfaces generated by the test set were perturbed by the device stochasticity.

The computation time for this test was heavily dependent on the number of clusters and the number of files. For this reason, we set the number of clusters to $N^{[1]} = 32$ for layer 1 and $N^{[2]} = 64$ for layer 2. We tested only on a random selection of 10% of the test set every run, and averaged performance over 60 runs.

To ensure the generality of our results, we included results using both the Support Vector Classifier (SVC) and the Euclidean distance classifier (Eucl.). Thus, we considered four cases: Ideal SVC, Noisy SVC, Ideal Eucl. and Noisy Eucl. Each network was trained on the same training sets of N-MNIST data and tested on the same randomly chosen N-MNIST test sets. We also measured classification performance for both layers of the architecture.

The results in Table 4 and Figure 6(a)) show that while different classifiers result in different absolute accuracy, stochasticity in the memristor dynamics does not significantly affect the classification accuracy (t-test $p > 0.05$).

However, device stochasticity does influence both time-surfaces and cluster assignment.

Fig. 6(b) compares time-surfaces computed over the entire array ($l = 28$) using the Ideal and Noisy models. Although the digit is still recognizable in the Noisy time-surface, we can clearly see additional variation in the individual pixels.

As shown in Figs. 6(c) and (d), this variation leads to incorrect cluster (i.e., polarity) assignment of Time-Surfaces, a phenomenon we refer to as 'dislocation' (Fig. 6(e). The mean percentage of events suffering from cluster dislocation in a single run was 10.58% in

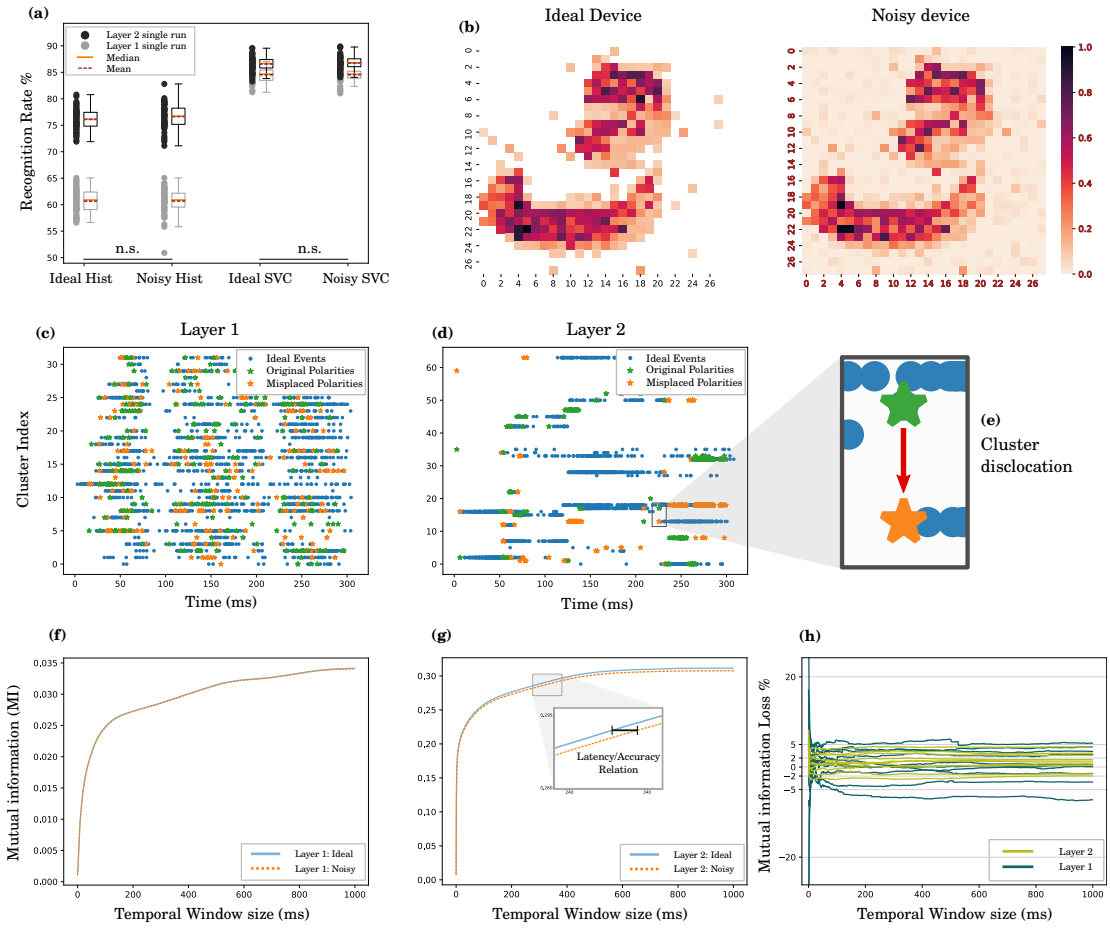

Figure 6: (a) Recognition rates of the Ideal and Noisy networks with the Support Vector and Euclidean Classifiers. (b) A qualitative comparison between Time-Surfaces computed over the entire input using the Ideal and Noisy memristor models. (c and d) The effect of Time-Surface perturbation on cluster (i.e., polarity) assignment in Layers 1 and 2. Blue dots indicate events where the Ideal and Noisy networks make the same cluster assignment. Otherwise, the Ideal (orange dots) and Noisy (green dots) assign events to different clusters. (e) We call this effect cluster dislocation. (f and g) The Mutual Information between the cluster response and the N-MNIST digit labels for Layers 1 and 2 at different Temporal Integration scales. (h) The Mutual Information Percentage Loss due to cluster dislocation. The effect of dislocation is small and constant across all timescales, except for a singularity when the temporal window is zero due to division by zero MI when computing the percentage. More importantly, cluster dislocation is equally likely to increase or decrease Mutual Information.

Table 4: Classification accuracy of the Ideal and Noisy Networks with Euclidean and SVC classifiers

	One HOTS Layer	Two HOTS Layers
Ideal Eucl	$60.67\% \pm 2.21\%$	$76.12\% \pm 1.98\%$
Noisy Eucl	$60.69\% \pm 2.33\%$	$76.66\% \pm 2.26\%$
Ideal SVC	$84.54\% \pm 1.37\%$	$86.56\% \pm 1.33\%$
Noisy SVC	$84.55\% \pm 1.33\%$	$86.76\% \pm 1.26\%$

Layer 1 and 7.09% in Layer 2, suggesting that multiple layers might decrease the noise effect.

One possible explanation for the maintenance of high classification accuracy despite cluster dislocation is that by summarizing events across the entire recording, histogram-based classifiers are "averaging out" the effect of a relatively small number of cluster dislocations. If this were true, then we might expect classifiers integrating information over shorter time scales to be far more affected by dislocation error.

To determine whether this is not the case, we calculated the Mutual Information (MI) between events at different time scales and labels, using a method originally presented by Akolkar et al.[3]. In this method, we choose a layer k, and sample a random event at that layer from our dataset ev_i^l. We create a temporal window of length δ centered on its timestamp t_i^k. We then look across different recordings to calculate the probability of finding another event with the same polarity p_i^k in the same temporal window. We can then calculate the MI between the probability of a response R (equal to 1 if an event with polarity p_i^k is present and 0 otherwise), $P(R)$, and the probability of the stimulus S (the label of the input), $P(S)$. We repeat this process multiple times including all polarities p_i averaging the result. This value tells us how well the spiking activity of the layer k at timescale δ encodes the dataset labels. We include additional information on this method in our Supplementary materials.

By computing the MI at different timescales δ, we can see how well spiking activity at different time scales encodes information about the stimulus labels. Since there is only one label for each recording in the N-MNIST and POKERDVS datasets, we expect the MI to

decrease monotonically as the timescale δ decreases, since shorter timescales contain less information (fewer spikes).

By comparing the MI computed for the Ideal and Noisy networks, we can see how cluster dislocation affects the MI. If it is true that classifiers integrating information over shorter timescales are more affected by dislocation error, then we would expect the Noisy network's MI to decrease faster than the Ideal network's MI as *delta* decreases.

Figs. 6(f-g) plot the MI for the Ideal and Noisy networks computed over ten runs using the N-MNIST dataset. As expected, the MI for both networks decreases as δ decreases. However, the two curves do not diverge as δ gets smaller, showing the introduction of cluster dislocation due to noise has little effect on the mutual information. We can show this more clearly using the Mutual Information Percentage Loss, defined by $(MI_{Ideal} - MI_{Noisy})/MI_{Ideal}$. Fig. 6(h) plots the Percentage Loss for each individual run. It remains largely constant across timescale, rarely exceeding 5% for the first layer and 2% for the second layer. In addition, the Mutual Information Percentage Loss is equally likely to be positive or negative 6(h). This suggests that rather than causing events to be mapped to less-informative clusters, cluster dislocation often results in events being mapped to equally, if not more, informative clusters.

The MI information loss might be considerably higher for different memristors or datasets. This could affect the accuracy of neuromorphic implementations. One possible solution suggested by this analysis is to exploit the monotonically increasing relationship between timescale and MI shown in Figs. 6(f) and (g), which indicates that the lost information might be recovered by increasing the time window size. However, the window size will negatively impact latency.

1.4.3 Computational Benefits Of Memristive Dynamics

In this section, we investigate whether the more complex dynamics of volatile LiWES memristors bring computational benefits compared to the simpler dynamics assumed in standard HOTS implementations.

We compare the classification accuracy of single-layer HOTS networks built with Ideal Memristor, a simulated Memristor without STP, and two traditional single-decay HOTS

architectures. In order to simulate a memristor without STP, we use the ideal model with Eq. (1-5,1-6), which causes the memristor response to reset to $\overline{A_1} + \overline{A_2}$ at each new incoming event ev_i after the end of the write pulse of width w has been reached. Additionally, we set a single exponential decay model by setting $k = 1$, obtaining the original single decay response without STP used for HOTS [51]. Each network was tested with 30 runs of the N-MNIST dataset. Both the training set and test set were independently sampled for each run.

(Fig 7) shows the results. For brevity, we only show results with the Euclidean classifier. Suppl. Table 3 contains additional results. The Memristor model is significantly more accurate than the Memristor without STP and the two single decay HOTS models. Enabling STP results in the largest increase in accuracy.

Our results also suggest that the double exponential decay better integrates temporal information. Figure 8 shows the effects of STP and double-exponential decays on the time-surface representations. Fig. 8(a) shows a full digit time-surface at a given time t_0 and a 11x11 Region-Of-Interest (ROI) with three distinct sub-regions. The ROIs are plotted in 8(b), with the sub-regions showing the model response to 'Recent events', 'Past events' and 'Sensor noise'. The last region represents a portion of the frame where the digit is not present. Activity is only caused by the typical salt and pepper noise of the DVS [58].

Fig. 8(c) shows the standard deviation of activity in the sub-regions, which is an indirect measure of the amount of information about recent, past, or noise events represented by the time-surface. Exponential decay kernels de-emphasize activity that is too fast or slow compared to their decay time-constants. This is evident when compare the standard deviation for recent events (light blue) in the HOTS Long Decay and for past events (dark blue) in the HOTS Short Decay.

In the original HOTS model [51], time-surfaces were computed using single exponential decay kernels. Thus, each layer is sensitive to activity only at a single temporal scale. Integration across multiple scales was obtained using multiple layers with increasingly longer time constants. However, the LiWES memristor has double exponential decay response with both short and long time constants. This enables a single layer to integrate information across multiple time scales simultaneously. Thus, the standard deviations of activity for recent and past events are comparable. This is true for both the Memristor model and the

Memristor without STP.

The standard deviation from the sensor noise region (in green) achieves its maximum for the HOTS Long Decay model and its minimum for the Memristor model. Longer delays cause the time-surfaces to accumulate multiple random events, increasing the standard deviation. In contrast, STP reduces standard deviation by summing the effect of multiple spikes, suppressing the effect of random events. This is consistent with our finding that the Memristor w/STP has smaller standard deviation in the sensor noise region compared to the Memristor w/o STP. This effect might also account for our finding that performance of the network is insensitive to device stochasticity.

1.5 Discussion

Currently, available neuromorphic processors are still in their infancy, as they aim to replicate biological neurons using silicon [45, 78, 15, 21]. However, their application has been limited due to several factors. First, our understanding of the brain is incomplete, lacking a comprehensive theory explaining its operations. Second, the different physical substrates of silicon and biological brains make it difficult to replicate the fundamental operation of temporal integration in the brain using neuromorphic architectures. Current solutions implement temporal integration digitally [20, 70] or through a combination of capacitors and transistors [62]. In contrast, this work utilizes an electrochemical memristor [110] with transitory conductance response to implement temporal integration on a single component, opening a path towards the development of compact and energy-efficient neuromorphic systems.

Advancements in technology offer a broader range of materials that could potentially facilitate the design of improved silicon-based brains. Architectures using this device challenge the conventional choices for abstraction level and partitioning in mixed-signal neuromorphic processors. These designs can employ more advanced computation building blocks and design rules. Determining the appropriate level of abstraction [38] remains an open question. The level of abstraction closely interacts with the physical substrate and the computational model design.

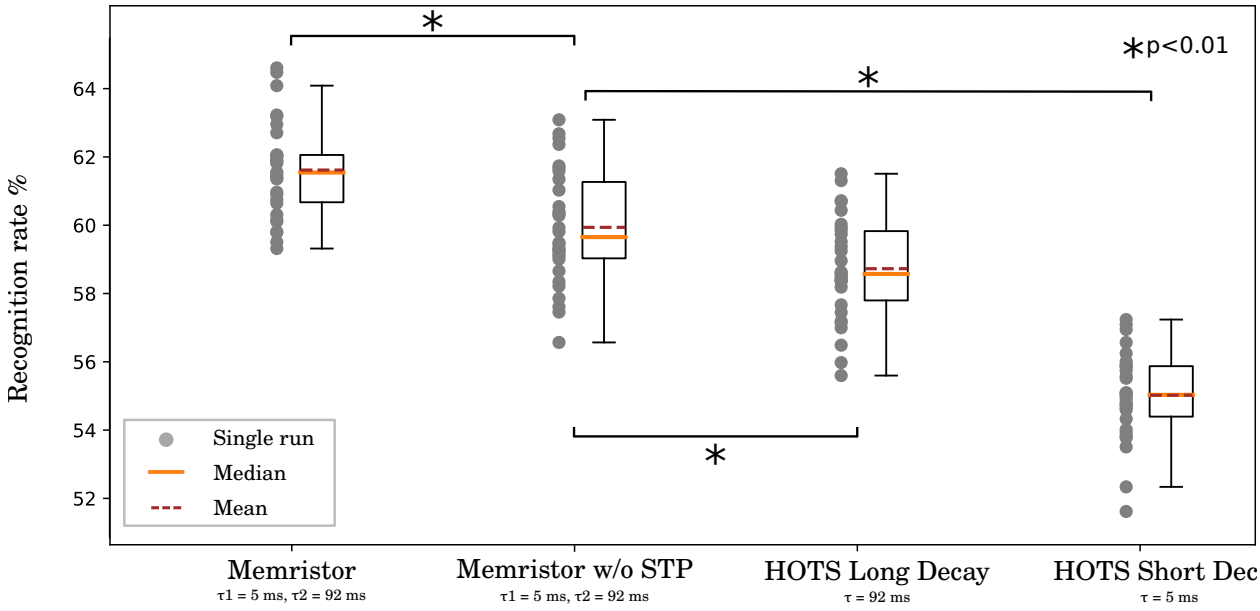

Figure 7: In this test, we compare the recognition rate of our memristor model (with STP and double exponential decay) against the same model without short-term plasticity (STP), HOTS with a single "long" decay, or a single "short" decay. Both double exponential decay and STP significantly ($p < 0.01$) improve accuracy over traditional single-decay w/o STP HOTS.

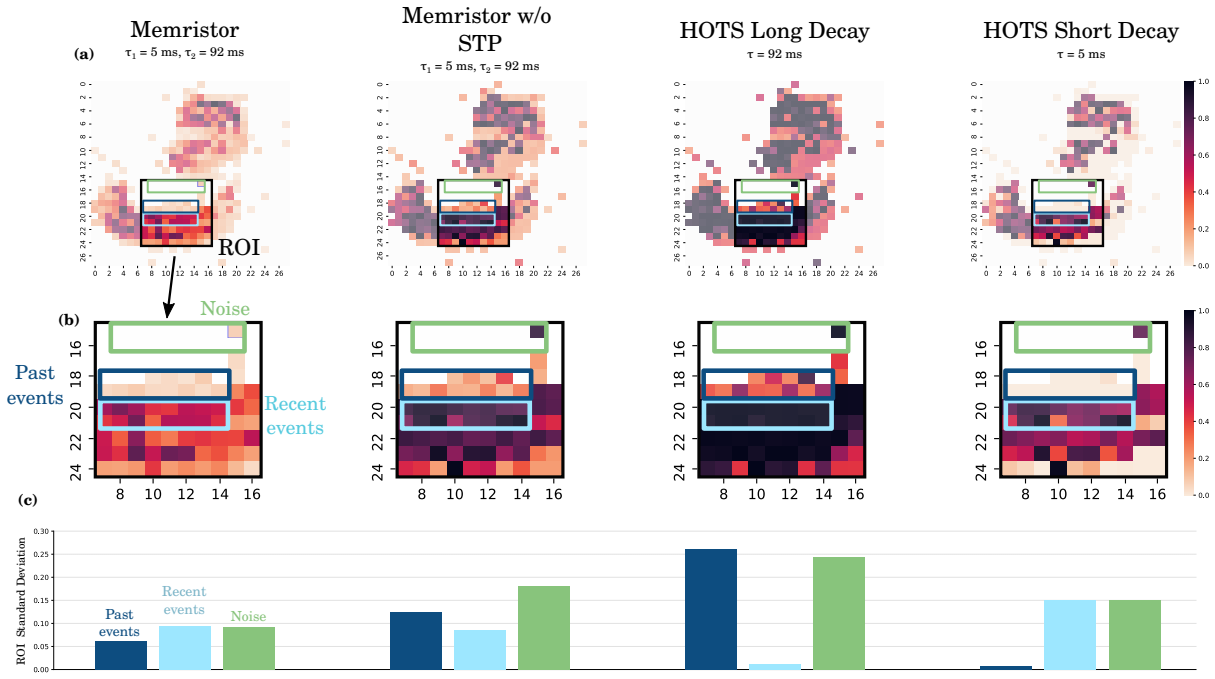

Figure 8: The effect of multiple exponential decays and STP on the Time-Surface representation. (a) time-surfaces for the same input computed using (left to right) HOTS with Ideal Memristor model, HOTS with the Ideal Memristor model but without STP, and the original HOTS with time constants of 5ms and 92ms. (b) Zoom in views of the 11x11 Region Of Interest (ROI), where we identify three sub-regions: recent events (light blue), previous events (dark blue), and sensor noise (green). (c) Plots of the standard deviation of activity in the three regions, which we use as a measure of information and noise. Because of STP, the Memristor model scores the lowest Standard Deviation in the sensor noise region (green). Networks with the double exponential decay (both for the Memristor model and the Memristor w/o STP) have similar standard deviation in windows corresponding to Past and Recent events. In contrast, HOTS single decay models can only represent information at a single timescale, resulting in near zero standard deviations for past events with short decays and for recent events for long decays.

The commonly used level of abstraction, which closely resembles direct biological replication, models neural computation using coupled ordinary differential equations. The temporal dynamics of this model enable information integration over time, while the coupling across state variables models spatial information integration [1]. Analog continuous time VLSI circuits, such as those described by Mead [60], are commonly used to implement this level of abstraction. Although these circuits offer low power consumption, they suffer from drawbacks such as mismatch and limited programmability.

A related level of abstraction involves coupled difference equations and is commonly implemented using digital design methodologies in standard CMOS processes or FPGAs [45, 78, 15, 21]. However, even with fully custom designs, these implementations fail to achieve the low power levels sought by neuromorphic engineers.

Hybrid substrates present an intriguing design space that can leverage the advantages of both analog and digital domains [100]. The most prevalent hybrid model utilizes analog circuits for computation and digital circuits for communication [84, 16, 44]. This approach recognizes that digital circuits operate much faster than the typical spike rate of neurons, enabling a single digital bus to carry signals from multiple neurons. Multi-chip Address Event Representation (AER) networks [71, 28] embody this level of abstraction, where computation within each chip utilizes analog continuous time circuits, while communication between circuits is digital and often asynchronous.

In this context, we advocate for a hybrid model that employs a different partitioning in the abstraction. Rather than dividing along the lines of function (computation vs. communication), we propose a partitioning based on dimension (time vs. space). We argue that analog implementation is optimal for temporal integration of signals, particularly spiking signals, while digital technologies are better suited for spatial integration. Newly developed memristive technologies [67, 121, 93, 102] provide an excellent physical substrate for temporal integration. In contrast, spatial integration, which requires signal communication across space, is best achieved using digital technologies.

The memristive network we study moves in the direction of the proposed partitioning. It exhibits space-time separability, as it integrates information over time, pixel by pixel, or more generally, neuron by neuron, followed by spatial integration across pixels or neurons.

Space-time separability is a well-known principle in digital signal processing algorithms, offering significant implementation advantages [37].

However, not all neuromorphic algorithms exhibit space-time separability. For instance, not every set of coupled differential equations can be expressed as a space-time separable set of operations, with the majority being unable to do so.

Nevertheless, we argue that a large and compelling class of algorithms, operating at an abstract scale without relying on spiking neurons, specifically those utilizing time-surfaces and hierarchies of time-surfaces (HOTS) [51, 97, 35], naturally exhibit space-time separability as described earlier. These algorithms are ideally suited for implementation using a combination of memristive devices for temporal integration and digital spatial integration, particularly clustering and mapping to the nearest cluster centers. By combining HOTS with a novel three-terminal memristor (Li_xWO_3), we illustrate how such architectures can be employed for pattern recognition while remaining robust against non-idealities encountered in memristive devices, such as random mismatch and noise.

1.6 Conclusion

Recent developments in semiconductor technology have led to the design and creation of a new class of devices called memristors. It has been predicted that memristors will be used in the near future as the atomic component of more advanced and complex systems, which can provide performance superior to conventional transistor-based hardware [101]. In neuromorphic engineering, memristors are more commonly used as "static" synaptic weights for the spatial integration of signals. While the temporal dynamics of memristors are known in the literature, we still need to understand their computational properties better to be able to exploit them fully in practical scenarios.

In this work, we used a Li_xWO_3 electrochemical memristor to test the effect of programmable time constants, double exponential decay and STP on the widely used N-MNIST dataset and POKERDVS dataset. We used single-pulse recordings to build a model of the device and then used it to simulate a HOTS network. The ability to program time constants

is important, as different tasks generate spike activity with different temporal dynamics as evidenced by our results comparing the best pulse parameters for the M-NIST and POKERDVS datasets. In this work, we assumed that time constants for all memristors in the network had the same statistics, but moving forward, it may be interesting to investigate setting time constants layer by layer or even neuron by neuron. We showed that the intrinsic stochasticity of the device did not impact accuracy (VI.B section). However, we also showed a relationship between latency and accuracy that could be used to offset accuracy loss by increasing integration time in devices with less precise temporal dynamics. This is especially important as it extends our considerations to memristors other than the one we tested, such as two-terminal memristors that also exhibit STP [67, 102].

One limitation of our results is that we could not include device mismatch in our simulation, as we have yet to realize a LiWES array, which would allow us to characterize this mismatch. However, we have modelled cycle-to-cycle variation, which also gives rise to mismatch in time-surfaces, albeit over time rather than space. Nonetheless, our results showing robustness to this type of mismatch suggest that our network may also be robust to spatial mismatch. This is a promising avenue for future work.

The last section (VI.C) explored the computational properties of STP and the double exponential dynamics. Both STP and double exponential decay dynamics increased the accuracy of the network compared to the original HOTS network with single exponentials and no-STP. STP contributes to reducing "Noise" in the network. Multiple exponential dynamics allow temporal integration across a broader time scales. Since time-surfaces are based on an exponential decay kernel, akin to biological EPSP/IPSPs, we expect our results to generalize to a wider class of models, such as integrate and fire neurons. These results are of particular importance as they highlight the practical use of less explored properties of this class of memristive devices and allow us to envision a future where memristors are used for temporal data processing and synaptic weight, eliminating the need for more complex analog or digital circuits.

Taken as a whole, our work provides strong evidence that the volatile properties of memristors can become a powerful tool for building a more specialized class of neuromorphic systems while reducing design complexity.

2.0 Sup3r: A Semi-Supervised Algorithm for increasing Sparsity, Stability, And Separability In Hierarchy Of Time-Surfaces architectures.

This chapter focuses on a novel learning rule for HOTS I started developing while still collaborating with Feng's group. During this period, I got galvanized by the potential of memristive devices for implementing local learning rules while growing increasingly frustrated with HOTS problems and limitations. One of the fundamental limitations is layer-wise learning, in which every layer has to be trained separately via k-means or other clustering algorithms. This method is time-consuming, expensive memory-wise when performed offline (as all data has to be loaded in memory to perform a centroid update), and underperforming (compared to SNN trained with backpropagation).

However, clustering offers a different interpretation for training neural networks altogether compared to backpropagation, where the training process is categorized as function fitting. For its compositionality, clustering offers a natural implementation of continual and incremental learning, properties that could be extremely useful for applications where data is time-varying. This is often the case of biomedical data and, more specifically, neural recordings.

For these reasons I became interested in the search of a novel algorithm for end-to-end supervised clustering that could also prove the properties of continual end incremental learning on HOTS. Inspiration for this work came from insightful discussions with prof Carl R. Olson, Caroline Runyan, Himanshu Akolkar, and Andy Schwartz.

Since my time at Pitt ended prematurely due to lack of funds, this work was not concluded, but it is publicly accessible via arXiv [80] and will be completed in the following months. The following sections are the ones published on arXiv version 2. This work was authored by me, under the supervision of Himanshu Akolkar, and Ryad Benosman.

2.1 Introduction

Hierarchy Of Time-Surfaces is a neuromorphic algorithm used to extract features from patterns of events [51]. This is possible thanks to a type of representation called time-surface or time-vector, where events are interpolated by exponential decay kernels and collected to represent relative time differences between the activation of units in the network. Time-surfaces are one of the most common representations in the neuromorphic field since they allow to interface event data with traditional machine learning and computer vision algorithms [31, 33]. In HOTS, time-surfaces are clustered together using algorithms like k-means to extract common activity patterns, and layers of units are built by considering each centroid as a neuron that can emit a new event when an input time-surface is assigned to it. For this reason, HOTS shares many points in common with bag-of-words or bag-of-features algorithms[19]. For instance, HOTS requires an external classifier on histograms of features to classify information. Similarly to bag-of-words algorithms, HOTS classifiers are histograms that accumulate features over a given temporal window to produce an input vector to traditional machine learning algorithms like Support Vector Machines and Multi-Layer Perceptrons[106, 81, 51, 82]. This approach limits compatibility with neuromorphic hardware and can nullify latency and energy efficiency advantages that are found in neuromorphic systems. Compared to Spiking Neural Networks (SNNs) trained with backpropagation through time, HOTS lags in accuracy [97, 34, 95, 51]. However, feature engineering and model tweaking can often bridge the gap on selected tasks [34, 97]. This is not surprising as it is known that clustering algorithms can reach comparable accuracy to small Convolutional Neural Networks (CNNs) when carefully tuned [17]. Moreover, clustering-based algorithms like HOTS are still worth investigating because of properties that might be inherited from clustering, like better explainability [69, 47] compared to more complex SNNs and continual and incremental learning[9, 75, 52]. This work presents Sup3r, a Semi-Supervised algorithm for increasing Sparsity Stability and Separability in the Hierarchy of Time-Surfaces. This algorithm can train end-to-end HOTS networks online, ditching the classifier for a Deep Neural Network architecture where every last unit encodes for a different label. We show that Sup3r can learn class-informative patterns and rejects events from confounding fea-

tures, reducing the number of processed events. Moreover, Sup3r allows the HOTS network to adapt to distribution shifts in the data (continual learning) and learn new tasks without forgetting (incremental learning). Finally, preliminary results on N-MNIST show that Sup3r can reach comparable accuracy to a similarly sized Artificial Neural Network (ANN) trained with back-propagation.

2.2 Time-surfaces And Time-Vectors

HOTS originally introduced the concept of time-surfaces [51]. These descriptors encode relative timings between events of units within a defined neighborhood and were historically used on events generated by event-based cameras [90]. This type of camera does not output frames but an asynchronous stream of events generated by individual pixels detecting brightness changes. Thus, in the context of vision, an event i can be described as the tuple:

$$ev_i = (x_i, y_i, p_i, t_i) \tag{2-1}$$

where t_i is the timestamp of the event, x_i and y_i are the horizontal and vertical coordinates of the pixels generating the event, and $p_i \in [0, 1]$ is the polarity of the event, which is 0 when the event signifies a decrease in brightness for the given pixel and 1 when it signifies an increase. To generalize to different modalities and uses, we can redefine events as:

$$ev_i = (\mathbf{x_i}, p_i, t_i) \tag{2-2}$$

where $\mathbf{x}$ is now a vector (in **bold**) representing any possible space coordinates depending on the event-based sensor type. In the case of events generated by event-based cameras $\mathbf{x} = (x, y)$ [51], whereas in the case of neuromorphic cochleas $\mathbf{x} = ch$ representing the channel index [81]. To generalize, we consider $p_i \in \mathbb{N}$ as we now use it to index events produced by multiple units at the same spatial location [51, 81]. Time-surfaces are built by centering a squared window R_i of lateral size l around every $\mathbf{x_i}$ and collecting the last event emitted by neighboring pixels in a time context T_i:

$$\mathbf{T_i} = \max_{j \leq i} \{t_j | \mathbf{x_j} \in R_i\} \tag{2-3}$$

Finally, time-surfaces are built by applying an exponential decay kernel on the time context:

$$\mathbf{ts_i} = e^{-\frac{(t_i - \mathbf{T_i})}{\tau}} \tag{2-4}$$

time-vectors $\mathbf{tv_i}$ are a more general definition of time surface where the number of dimensions is one or more than two [81]. In HOTS, time-surfaces are clustered with clustering algorithms like k-means. Whenever a time-surface is assigned to a centroid, it produces an event. These events can be used to generate new time-surfaces that can be clustered with longer taus to integrate and extract more complex patterns of activities [51]. Events outputted by a layer k are so defined:

$$ev_i^{k+1} = (\mathbf{x_i}^{k+1}, p_i^{k+1}, t_i^{k+1}) \tag{2-5}$$

where $\mathbf{x_i}^{k+1} = \mathbf{x_i}^k$, $t_i^{k+1} = t_i^k$, and $p_i^{k+1} = f_i^k$ with f being the "firing" feature/centroid assigned to the input time-surface ts_i^k.

2.3 A Hierarchy Of Time-Surfaces And Time-Vectors

A fundamental principle behind HOTS networks is the increase of spatial neighborhood R^k and τ^k at every layer to extract longer and more complex spatial and temporal patterns. In this work, we use the same principle, with the difference of the last layer K (where K denotes the last layer of a network with layer index k) where R^K is always including all coordinates $\mathbf{x}$. This is somewhat different from traditional HOTS implementation, where it is often the external classifier that integrates all the possible spatial coordinates.

2.4 Sparsity, Separability, And Stability In Spiking Networks

Sup3r aims to remove external classifiers from HOTS by using the network's last layer as a classifier for a pattern recognition problem. In that case, we expect the number of clusters to equal the number of classes in the problem and for each cluster to generate events only when the correct class is presented to the network. In this context, the classifier will have to limit the firing to only one unit (high **Sparsity**) for the whole duration of the stimuli (high **Stability**). In the context of clustering, this also means that classes assigned to different centroids must be well **Separated**. Conveniently, we can represent these three concepts using a time-vector called feedback-time-vector $\mathbf{ftv}^K$, built on the output events ev_i^{K+1} of our classifier. Rather than pulling the last event in a given neighborhood R (eq. 2-3), we pull the last event of every polarity in the last layer:

$$\mathbf{FT_i^K} = \max_{j \leq i} \{t_j^{K+1} | \mathbf{p_j^{K+1}}\} \tag{2-6}$$

We use $\mathbf{ftv}^k$ to differentiate feedback time-vectors from standard time-vectors $\mathbf{tv}^K$ or time-surfaces $\mathbf{ts^K}$ built on the input events ev_i^K. We then use it to build the descriptor S:

$$S_i^K = G(\mathbf{ftv}_i^K[f] - \sum_{l \neq f}^{N^K} \frac{\mathbf{ftv}_i^K[l]}{K-1}) \tag{2-7}$$

Where f is the index of the feedback-time-vector element corresponding to the "firing" centroid (last event) that triggered its generation, and N^K is the number of clusters for layer K. G is a sign function that is positive if the "firing" centroid corresponds to the correct class label, negative if otherwise:

$$G = \begin{cases} + & \text{for } f = cc \\ - & \text{for } f \neq cc \end{cases} \tag{2-8}$$

where cc is the index of the correct class. To see how this descriptor relates to Sparsity, Stability, and Separability, let us look at the limits for S. If the classifier has a stable,

sparse, and correct response (only one unit active, corresponding to the correct class index cl), S will tend to 1:

$$S_i^K = +(1 - \sum_{l \neq f}^{N^K} \frac{0}{K-1}) = 1 \tag{2-9}$$

Conversely, if only one unit is active, but it corresponds to the wrong class label, G will be negative:

$$S_i^K = -(1 - \sum_{l \neq f}^{N^K} \frac{0}{K-1}) = -1 \tag{2-10}$$

indicating a fully sparse and stable but wrong response, meaning the classifier does not correctly separate the classes.

In case the activity of the classifier is dense, meaning that all clusters are firing events at approximately the same time, the feedback time-vectors will become fully populated by ones, making $S = 0$:

$$S_i^K = G(1 - \sum_{l \neq f}^{N^K} \frac{1}{K-1}) = G(1 - \frac{K-1}{K-1}) = G(0) = 0 \tag{2-11}$$

Thus, S amplitude $\in [0, 1]$ represents the degree of sparsity and stability of the last layer activation, where 1 denotes a fully sparse and stable response, and 0 indicates a dense and unstable activity. We can extend this descriptor to every layer of the network k to characterize the degree of Sparsity, Stability, and Separability of all the centroids in multi-layer networks. The only difference with layer K is that intermediate layers will output events with spatial coordinates since they are not integrating over all coordinates $\mathbf{x}$ (sec. 2.3). Therefore, the feedback time context FT_i^k will be defined as such:

$$\mathbf{FT_i^k} = \max_{j \leq i} \{t_j^{k+1} | \mathbf{x_i^{k+1}}, \mathbf{p_j^{k+1}}\} \tag{2-12}$$

meaning that $\mathbf{FT_i^k}$ pulls events with different polarities $\mathbf{p_j^{k+1}}$ and the same spatial coordinates $\mathbf{x_i^{k+1}}$.

2.5 Learning Class-Relevant Features

Since we want to develop an online and local learning rule, we cannot resort to backpropagation to maximize S, especially if we consider that HOTS neurons are not differentiable. Therefore, we resort to a different approach, focusing on using a modified online k-means learning rule. The learning rule we propose is described by the following equation:

$$\Delta \mathbf{c}_f^k = \alpha \Delta S_i^{k+1} \mathbf{q}_{i,f}^k + \beta S_i^{k+1} \mathbf{q}_{i,f}^k \tag{2-13}$$

where α andβ are learning rates, $\mathbf{c}_f^k$ represents the position of the "firing" centroid, assigned to a novel time-surface $\mathbf{ts}_i^k$ and $\mathbf{q}_{i,f}^k = \mathbf{ts}_i^k - \mathbf{c}_f^k$. If we assume $S_i^k, \Delta S_i^k = 1$ and $\eta = \alpha + \beta$, the equation takes the form of online k-means [9]:

$$\Delta \mathbf{c}_f^k = \eta \, \mathbf{q}_{i,f}^k = \eta \, (\mathbf{ts}_i^k - \mathbf{c}_f^k) \tag{2-14}$$

As shown (eq: 2-13), the proposed learning rule comprises two terms that weight the online k-means equation (eq: 2-14) by ΔS and S. The first term allows centroids to only move in the direction of a set of features when they are associated with an increase in S. Conversely, centroids will move away from features that decrease S (such as features of the wrong class or class-unrelated features), ensuring that only class-relevant features are learned. As $S \to 1$, $\Delta S \to 0$ that makes the first term of equation 2-13 tend to 0. This ensures the learning rate will decrease as we approach a maximum, but it can cause the centroids never to reach the center of mass of clusters. For this reason, the second term of the learning rule is weighted by S but with a different learning rate $\beta < \alpha$ to ensure it is not adding instability to local maxima.

2.6 Individual Thresholds

One problem arising from the presented learning rule is the choice of the decision boundaries for the cluster assignments. In k-means, the decision boundaries are defined as the

equidistant points between two adjacent clusters, similar to the borders of a Voronoi diagram. Inconveniently, the learning rule presented with equation 2-13 is not compatible with this type of decision boundary. The reason is that this type of segmentation will always assign every sample to a centroid, making it impossible to learn only class-relevant features. This means that when some centroids are pushed to more relevant features, the less relevant features will be assigned to other nearby clusters, reducing their specificity and ultimately lowering classification accuracy. To avoid this scenario, we introduce individual thresholds th_n^k, where n is the cluster index for layer k. We initially assign every sample to the closest cluster as done in k-means, but now, every cluster will generate a new event only when the sample distance to the cluster is less than th_n^k. When this happens, we tune individual thresholds with the equation:

$$\Delta th_f^k = \gamma \Delta S_i^{k+1} e^{-\frac{\|\mathbf{q}_{i,f}^k\|_2}{d^k}} + \delta S_i^{k+1} e^{-\frac{\|\mathbf{q}_{i,f}^k\|_2}{d^k}} \tag{2-15}$$

where d is a parameter used to control how close to the border samples need to be to have an effect on the threshold update. We also update nearby thresholds with a competitive rule to reduce the number of events produced by the network and threshold overlap between adjacent clusters. If a sample falls within the decision boundaries of other "non-firing" clusters, represented by the index nf, and ΔS and S are positive, we update their thresholds with the following equation:

$$\Delta th_{nf}^k = -\gamma \Delta S_i^{k+1} e^{-\frac{\|\mathbf{q}_{i,nf}^k\|_2}{d^k}} - \delta S_i^{k+1} e^{-\frac{\|\mathbf{q}_{i,nf}^k\|_2}{d^k}} \tag{2-16}$$

Similarly to equation 2-13, both rule presents two terms: one weighted by ΔS_i^{k+1} and one weighted by S_i^{k+1}. The first term changes the decision boundaries to include samples associated with an increase of S and reject samples associated with its decrease. In the situation where the network is consistently wrong or consistently right, this first term will tend to 0, requiring a second term weighted by S_i^{k+1} to increase network firing rate and stability for correct classes and reduce them for incorrect ones. As noted for equation 2-13, $\delta < \gamma$ to ensure the stability of the learning rule.

2.7 Network Initialization

Similarly to k-means, centroid initialization is an important step to ensure good results [36]. Random initialization might put the centroids too far away from the feature spaces or regions where the sum of updates from the learning rule might push centroids even further from any meaningful features. However, we are not interested in state-of-the-art initialization techniques since they might obfuscate the actual performance results of Sup3r. Therefore, we initialize centroids on the average time-surfaces from a small batch of data and add noise drawn from a uniform distribution:

$$\mathbf{c_j^k} = (1 - \zeta)\,\overline{\mathbf{ts^k}} + \zeta\,\mu_{ts}\,\mathbf{U_j^k} \tag{2-17}$$

Were $\mathbf{U}$ represents a matrix with the same shape as $\mathbf{c_j^k}$ and composed of random values pulled from a uniform distribution. Since $\mathbf{ts_i}$ with small τ are mostly 0 while $\mathbf{ts_i}$ with high τ are mostly 1, we scale $\mathbf{U}$ by the average of the elements in $\overline{\mathbf{ts^k}}$, defined as μ_{ts}. This approach is extremely simple. In the future, more complex approaches could be explored to ensure the convergence of most features and boost learning speed and accuracy.

2.8 Results

2.8.1 Synthetic Benchmark

Hierarchy of Time-Surfaces uses unsupervised clustering algorithms such as online k-means to extract recurring features in the data. While this approach can be effective for pre-training on large quantities of unlabeled data, it presents several drawbacks compared to state-of-the-art neuromorphic networks that use approximate gradient methods based on Back Propagation Through Time. One problem is the need for an external non-neuromorphic classifier, which we outlined in the introduction. A second problem is the loss of accuracy when learning non-class-relevant features. To best represent this problem, we design a tailored neuromorphic synthetic benchmark. We propose a visual pattern classification task

requiring a three-layered network (fig. 9). The proposed task is to classify two "sentences" ("v/v yty" and "vxv yty") composed of two "words" each made of three "characters" drawn by lattice of 5x5 Poisson spiking neurons, for a total of 5x30 neurons (as can be seen in fig. 9a). Yellow pixels correspond to "high" firing rate while blue pixels correspond to "low" firing rate. We use Poisson neurons and background firing rate to introduce variability per every sample. The training set consists of 1000 different recordings (500 for each sentence) of 10ms. This visual pattern is then classified by a three-layered convolutional HOTS neural network (fig. 9b) where the first layer ($N^1 = 3$) responds to individual characters, the second layer ($N^2 = 6$) responds to single words, and the last layer ($N^3 = 2$) classifies the entire sentence. It should be noted that the number of HOTS centroids for each layer (N_i) does not correspond to the actual number of clusters in the data. In the first layer, the number of clusters is five (all the possible characters: "v, /, x, y, t"), while in the second layer, this number is three (all the possible words in the dataset: "v/v, vxv, yty"). This choice is motivated by the need to represent real-world scenarios in which the number of clusters is unknown. This can be especially problematic when the number of centroids is less than the actual number of clusters in the data, as similar clusters representing two features of distinct classes might be assigned a single centroid. This is the case of layer 1, making it likely for the same centroid to represent similar characters like "x" and "/". These two characters are the only difference in the two sentences in the dataset (fig 9a), making them class-relevant features. Failure to assign separate centroids to "x" and "/" means that layer 1 will respond similarly to the two sentences, causing the other network layers to fail to separate the two sentences.

2.8.1.1 Classification Results

We show the classification results of a separate test set of 1000 recordings (500 recordings for each sentence). We test Sup3r against a comparable network with k-means features. Figure 10 shows the features from layer 1 for both architectures. As expected, Sup3r can leverage Separability Sparsity and Stability from higher layers to extract "x" and "/" (fig. 10a) without utilizing the third centroid. Oppositely, k-means must assign every centroid

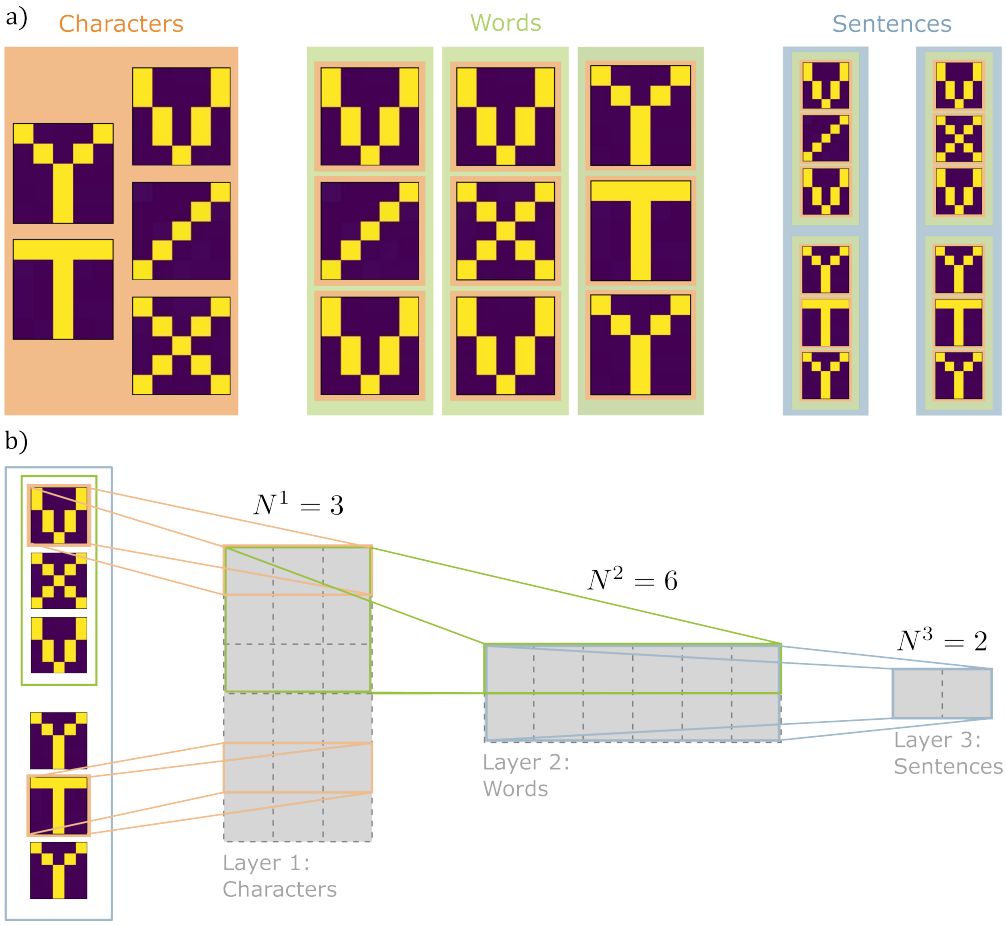

Figure 9: The visual classification benchmark to test Sup3r on a small 3-layered network. a) The test is composed of a "sentence" (blue) classification task, where two different sentences are each composed of two among three different "words" (green), which are composed of three of five different "characters" (orange). The only difference between the two sentences consists of two characters, "x" and "/", which are contained in the top word. To generate spiking data, sentences are written by a lattice of 5x30 Poisson neurons firing at a high firing rate for the yellow pixels and a low firing rate for the blue pixels. b) The convolutional HOTS network used to solve the task. The gray rectangles represent the output dimensionality of each layer. N is the number of clusters responding to the six different characters in a sentence (Layer 1), to the two words in a sentence (Layer 2), and finally to the entire sentence (Layer 3).

to a group of samples and minimize their variance, an optimization technique that forces similar features to share a centroid when the number of centroids is below the number of actual clusters in the dataset. In figure 10b, we define the network accuracy by calculating the percentage of events assigned to the right sentence in a single recording. The k-means algorithm causes HOTS to fail to recognize the two sentences, resulting in ≈55.36% accuracy (close to the chance level). Conversely, Sup3r reaches ≈99.92% accuracy while using only two centroids. Since Sup3r can reject any event produced by the non-class-relevant features, it outputs only ≈14.33% of the input events (fig. 10c), potentially reducing energy consumption in hardware implementations. To train the network we used the following parameters: $\alpha = 1 \cdot 10^{-4}$, $\beta = 1 \cdot 10^{-5}$, $\gamma = 1 \cdot 10^{-4}$, $\delta = 5 \cdot 10^{-6}$, $\tau_1 = 1s$,$\tau_2 = 1ms$, $\tau_3 = 1ms$ and **ftv** taus $f\tau_2 = 100ms$ and $f\tau_3 = 10ms$. We also report the results in table 5.

2.8.1.2 Continual Learning

Sup3r learning signal S_i^K is composed of a local signal (coming only from the next layer) and a global signal G. It is important to note that the local signal is entirely unsupervised, as there is no need to know the correct label to calculate it (eq. 2-7). This opens the possibility of keeping the unsupervised learning running while inferring to adapt to potential shifts in the class distributions. To test this feature, we design a test set of 1750 recordings by shifting "x" and "/" characters in the test sentences. To shift the characters, we subtract pixels from "x" to add them to "/" incrementally, with the final shift causing the two letters to swap entirely (fig. 11a). Every shift lasts for 250 recordings, with the last shift lasting for 1000 to ensure convergence and stability. Looking at the Sup3r centroids, we can see them adapting to the shift so that "x" and "/" become swapped (fig. 11b). To evaluate Sup3r accuracy, in figure 11c, we test the network against an ablated network with no unsupervised learning. As expected, Sup3r adapts to distribution shifts while maintaining the original accuracy (>99%), while the ablated network fails the task, ending with 0% accuracy. This final test was performed with 10 independently trained networks with 10 distinct test sets. To train the network, we used the same parameters we used for the classification results.

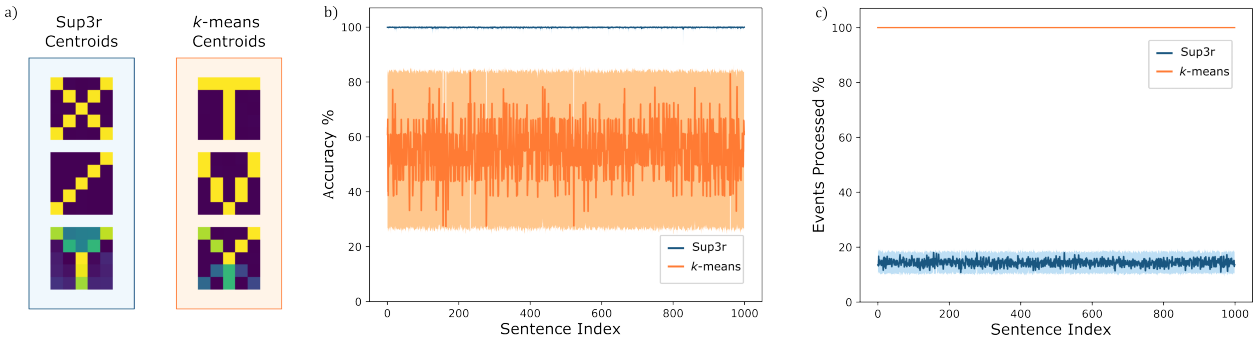

Figure 10: Neuromorphic benchmark results, comparing Sup3r against k-means for feature extraction. a) Layer 1 features extracted with k-means (light blue) and Sup3r (orange). k-means fails to extract class-relevant features, while Sup3r can correctly extract "x" "/", leaving one centroid unused. b) Consequently, k-means fails the classification task, performing close to the chance level (≈55.36% accuracy, calculated as the percentage of events assigned to the correct class), while Sup3r can solve the task with ≈99.92% accuracy. Since Sup3r rejects events from non-class-relevant characters, only ≈14.33% of events are propagated in the network (calculated as the ratio between the number of the last layer output events divided by the number of the first layer input events). Conversely, the k-means network outputs the same number of input events.

Table 5: Classification results (Mean on 10 networks trained and tested on independently generated synthetic datasets)

	Accuracy	Processed events
Sup3r Network	99.92%	14.33%
k-means	55.53%	100%

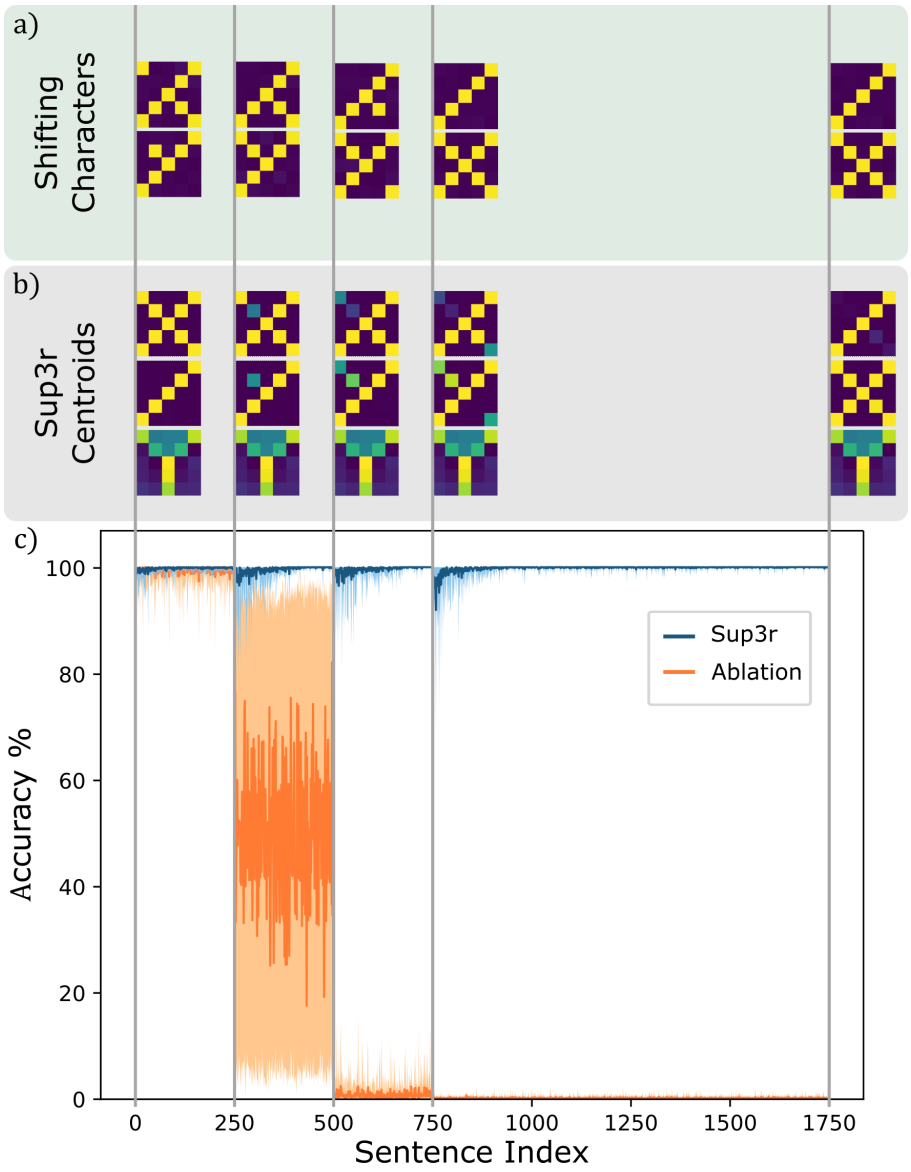

Figure 11: Continual learning test with Sup3r. In this test, we perform a class shift in the two sentences by subtracting pixels from "x" and adding them to "/" every 250 sentences (a), with the final shift lasting 1000 sentences. Sup3r centroids slowly adapt to the shift (b). In (c), Sup3r network accuracy remains high and returns to ¿99%, while the same network without unsupervised learning fails the task (Ablation). This test is performed with ten independently trained networks and randomized test sets. Accuracy is reported as the average across runs in dark colors with min and max values in light colors.

Table 6: Incremental Learning results (Mean on 10 networks trained and tested on independently generated synthetic datasets)

	Task 1	Task 2	Task 1 after Task 2
Sup3r Network	99.92%	99.99%	99.96%

2.8.1.3 Incremental Learning

Our hypothesis is that Sup3r ability to extract sparse class-relevant features makes it a natural candidate for Incremental Learning. Since we are extracting a set of sparse class-relevant features, Sup3r might learn new tasks simply by adding new centroids and fixing the previous ones to avoid catastrophic forgetting.

To test this hypothesis, we design a new task (task 2) in which a Sup3r network has to learn two novel sentences, "vpv yty" and "vov yty", after being trained on the previous classification task (task 1) in section 2.8.1. We then tested the same network on task 1 after retraining on task 2 to demonstrate that it did not suffer catastrophic forgetting. We show the results in Table 6.

The results show that the network can train on Task 2 and preserve the accuracy previously reached on Task 1. To train the network, we used the same parameters we used for the classification results.

2.9 Sup3r Against Backpropagation

We present preliminary results to show how Sup3r compares against backpropagation. We use the N-MNIST dataset [68] a neuromorphic version of the popular MNIST dataset [24]. This allows us to compare a Sup3r-HOTS network with a traditional ANN. The Sup3r HOTS network has a single hidden layer with 32 centroids and spatial neighborhood $R^1 = 9$. The ANN has the same architecture with a single convolutional hidden layer with 32 units with sigmoid activation and filter size 9x9. We train the ANN with stochastic gradient

Table 7: Sup3r NMNIST comparison

	Sup3r on N-MNIST	ANN on MNIST	HOTS on N-MNIST[82]
Test Accuracy	94.98%	95.61%	91.27%
% Processed Events	72.61%	N/A	100%

*HOTS implementation consisting of 2 layers (32-96 centroids) of double exponential time-surfaces and an SVM polynomial classifier [82].

descent and a learning rate of 0.05. Other Sup3r parameters where $\tau_1 = 100ms$, $\tau_2 = 1ms$, $\alpha = 5 \cdot 10^{-3}$, $\beta = 5 \cdot 10^{-6}$, $\gamma = 1 \cdot 10^{-4}$, $\delta = 1 \cdot 10^{-7}$. The reason for such low learning rates was the batch-based training implemented on GPU via OpenCL to achieve higher parallelism and reduce computation time. In this implementation, Δc_f^k are summed for every event, causing large centroid and threshold updates, which are compensated by lowering the learning rate. Table 7 shows the result of this comparison.

Sup3r accuracy is comparable to an ANN on MNIST and higher than HOTS best results [82] on a two-layered network (32-96 centroids) with double exponential decay time-surfaces and a polynomial SVM classifier, while cutting off 28% of events. While these results are promising, it is important to underline that they are preliminary. The reason is that this test was implemented on OpenCL for GPU acceleration, prioritizing code readability and debugging rather than hardware optimization. This limited the number of experiments we run on N-MNIST. A single epoch of the N-MNIST dataset runs between 2 hours and 45 minutes on an NVIDIA 2060 Super, depending on the number of processed events in the network, as computation is done event by event. This is due to the high CPU usage to queue multiple kernels per event, the much bigger N-MNIST dataset compared to his frame-based counterpart (several thousands of sparse events compared to a single 28x28 frame), and to HOTS, which requires several small matrix operations like multiplications and accumulation to process a single event. Exploiting vendor-specific optimized kernels, sparse libraries, kernel launching from GPU, and matrix operations should significantly reduce computation time and allow for an exhaustive hyperparameter analysis and testing on more complex

architectures and datasets.

2.10 Conclusion

In this paper, we presented Sup3r, a novel learning algorithm for training Hierarchy of time-surfaces algorithms. Our results show that this algorithm can extract class-relevant features, reduce the number of processed events, and enable incremental and continual learning without the need for external classifiers. Moreover, preliminary results show that this algorithm could reach comparable accuracy to backpropagation on a small network. All the results together make Sup3r the best way to train HOTS networks end-to-end. Future works will have to focus on accelerating Sup3r on GPUs, testing it on more complex datasets and deep-layered architectures, and reducing the excessive number of hyperparameters. Another point of interest could be to adapt Sup3r to work on much more popular Spiking Neural Network models like Integrate and Fire neurons.

3.0 Conclusions

This thesis focused on fundamental hardware and software challenges of future iBCIs neuromorphic decoders.

From a hardware perspective, the thesis explored the use of ECRAM memristor's volatile properties to implement temporal synapses and time-surfaces within a single device. This marks an incremental step for developing efficient and compact hardware decoders capable of processing spiking data with low latency and efficiency.

On the software side, this thesis proposed Sup3r, an online learning algorithm that enables Incremental and Continual learning on HOTS. On future neuromorphic decoders, this algorithm will enable adaptation to data distribution shifts to avoid re-calibrating and learning new tasks, resulting in a significant life improvement for iBCIs users.

However, further research is needed to develop a neuromorphic decoder based on Sup3r and ECRAM memristors. Primarily, we need to demonstrate a hardware implementation of a Spiking neural network on an ECRAM crossbar array using standard CMOS fabrication. Encouragingly, recent developments, such as the demonstration of the first crossbar array on CMOS technology [14], demonstrate the feasibility of this approach, although lacking any analysis of the device's volatile properties. Additionally, efforts must be directed toward mapping Sup3r-HOTS onto ECRAM crossbar arrays and designing specialized hardware blocks to implement the proposed learning rule and HOTS neurons.

On the software front, further work is needed to validate this decoder's efficacy through animal trials to fully characterize accuracy, latency, energy consumption, and robustness to shifts in recorded activity. Another possible research avenue is to generalize Sup3r to perform regression. This is necessary to enable fine movement decoding to control end effectors such as robotic arms.

In conclusion, while initial steps have been made, the journey towards realizing a fully functional neuromorphic decoder based on Sup3r and ECRAM memristors remains ongoing, with further research and experimentation required to bridge existing gaps and unlock the full potential of these technologies. In the future, such decoders will enable scalable, robust,

and efficient iBCIs, enabling patients to regain autonomy outside controlled environments and research labs.

Appendix Examples And Results

A.1 Testing LiWES, A Li_xWO_3 Electrochemical Synapse For Spiking Neural Networks

A.1.1 Model Fitting And Parameters Estimation

After defining the model, we need to estimate its parameters, namely, A_1, A_2, τ_1, τ_2. We apply a single voltage pulse at the Gate using a function generator (Tektronix AFG3252C). We then read the channel current I_{DS} by applying a small reading voltage bias (100 mV) between source (S) and drain (D) using a Keithley Semiconductor Parameter Analyzer (4200-SCS) at a sampling rate of 6 ms, no additional circuit is required. We calculate the device conductance using Ohm's rule; $G_{DS} = I_{DS}/V_{DS}$. This temporal precision is not enough to evaluate G_{rise}, as all the pulse durations are sub-millisecond. Therefore, we can only determine the parameters by fitting G_{decay}. Before fitting, we scale the data by removing the mean base conductance at the steady state, and we normalize by dividing the data by the mean maximum conductance over multiple recordings. After this step, we fit G_{decay} using the Gauss-Newton algorithm in the statistical analysis software JMP®. We can now obtain sets of all the necessary parameters from multiple recordings of the Li_xWO_3 synaptic memristor.

A.1.2 Mutual Information

Information was calculated using this equation:

$$I_\delta(S;R) = \sum_s P(s) \sum_r P(r|s)_\delta log_2(\frac{P(r|s)_\delta}{P(r)_\delta}) \qquad \text{(Supp. Eq: 1)}$$

with r being the response of the network, either a 1 or a 0, representing the presence of an event $ev_i^{[k]}$ with a given polarity p, and s being the "type" of the stimuli eliciting the response (the N-MNIST digit). In other words, $I_\delta(S;R)$ gives a measure of how reliably a single cluster of the network, tends to respond to the same type of input stimuli. Since

we used an equal number of different digits to calculate I_δ the probability of s is uniform $P(s) = 1/10$. For $P(r)_\delta$, we randomly extracted events $ev_i^{[k]}$ from single recordings, and then calculated the probability of finding an event $ev_j^{[k]}$ with the same polarity, in any other recording, within a time window of length δ centered around the time stamp t_i of the original event. Similarly to Akolkar et al. [3] responses were binary, so if multiple events with the same polarity were present within the time window of the same recording, only one was considered. To calculate $P(r|s)_\delta$, we repeated the process used for $P(r)_\delta$ but only searched events over multiple recordings of the same digit s. Finally, we calculated $I_\delta(S;R)$ over different time windows sizes (δ) and averaged the response for all the different polarities and multiple runs, defining a metric (MI) of how much information can be extracted from the network response in a given time window [Fig. 5(f,g)]. The results we show were calculated on the same recordings used for the individual runs in Fig. 5(a), in order to offer a true comparison between the recognition rate and mutual information. In order to validate the use of the proposed Mutual Information calculation as a proper metric for the stochasticity of the device and its effects on recognition rates, we simulated a hypothetical device with the same parameters used for the results shown in Fig. 5, but increasing the standard deviation of the model parameters by 2x, 5x, 10x. These results are reported in Supp. Fig 13 and show that both Mutual Information and recognition rate decrease for higher deviation in the device response (higher standard deviation in the stochastic parameters Eq. [6,7]).

A.1.3 Short Term Plasticity And Time-Surface Normalization

While STP could play a role in averaging and removing noise, it also presents a problem for clustering algorithms as k-means. In HOTS-like algorithms, Time-Surfaces are naturally bounded between 0 and 1; however, in this model, the memristor conductance is the sum of multiple events, which causes Time-Surfaces to reach values above 1. Unbound Time-Surfaces can impact classification, causing the clustering algorithm to perform poorly [17]. To avoid this problem, we normalized Time-Surfaces before clustering with k-means.

Table 8: Classification accuracy using memristors dynamics vs original exponential decay kernel of HOTS

	Avg. Accuracy ± Std. Deviation	t-test p-value (vs Memristor w/o STP)
Memristor w/o STP	$56.92\% \pm 1.73\%$	N.A.
2 Layers HOTS $N^{[2]} = 32$	$54.96\% \pm 2.19\%$	$3.52 * 10^{-7}$
2 Layers HOTS $N^{[2]} = 64$	$57.92\% \pm 1.84\%$	0.003

Results obtained with $N^{[1]} = 32$ clusters over 60 consecutive runs.

A.1.4 1 Layer Memristor Model Vs. 2 Layers Of Traditional HOTS

In addition to comparing a single memristor layer and a single HOTS layer, we also tested two layers of HOTS against a single memristor layer w/o STP (Suppl. Table 8). Interestingly, keeping the same number of clusters for both layers $N^{[1,2]} = 32$ causes a significantly lower accuracy than the double exponential decay of the Memristor model. This is probably due to the fact that clusters act as separate polarities for the next layer, increasing the distance between different patterns of different polarities when compared to different patterns from the same polarity. This effect might push the next layer to associate a single cluster for each polarity requiring more clusters to achieve a better feature extraction. A 2 Layer network from HOTS with $N^{[1]} = 32$ and $N^{[2]} = 64$ achieves similar performance to the Memristor model with $N^{[1]} = 32$ (Suppl. Table 8).

A.1.5 Additional Methods

All the simulations were performed in Python. Clustering was achieved using a mini-batch k-means algorithm as a faster and more efficient Python multithreaded implementation [88].

Table 9: N-MNIST classification accuracy with memristor dynamics using different pulse parameters

Pulse settings	(200μs,1V)	(200μs,2V)	(200μs,3V)	(200ms,4V)	(500μs,1V)	(750μs,1V)	(1ms,1V)
One Layer (Hist)	62.57%± 1.11%	63.45%± 2.29%	**64.11**%± 1.73%	63.68%± 1.16%	63.30%± 1.84%	61.30%± 0.84%	62.90%± 2.11%
Two Layers (Hist)	81.11%± 0.87%	82.01%± 0.64%	82.20%± 1.23%	81.91%± 0.38%	**82.21**%± 1.42%	80.36%± 1.60%	81.14%± 0.89%
One Layer (SVC)	90.86%± 0.51%	90.61%± 0.43%	90.86%± 0.34%	90.86%± 0.55%	**90.94**%± 0.46%	90.78%± 0.27%	90.79%± 0.39%
Two Layers (SVC)	90.90%± 0.22%	90.92%± 0.24%	91.15%± 0.37%	90.93%± 0.21%	90.09%± 0.16%	90.95%± 0.19%	**91.27**%± 0.29%

Results obtained with $N^{[1]} = 32$ clusters for layer 1 and $N^{[2]} = 96$ for layer 2 over 5 runs and with pulse characteristics from Table. 1,2 [Main text]

Table 10: POKERDVS classification accuracy with memristor dynamics using different pulse parameters

Pulse settings	(200μs,1V)	(200μs,2V)	(200μs,3V)	(200ms,4V)	(500μs,1V)	(750μs,1V)	(1ms,1V)
One Layer (Hist)	76.00%± 11.58%	80.00%± 8.94%	**84.50**%± 10.36%	74.50%± 15.72%	83.00%± 8.12%	76.00%± 12.81%	77.00%± 10.53%
Two Layers (Hist)	95.50%± 4.15%	96.00%± 4.36%	95.00%± 3.88%	**96.50**%± 3.20%	95.50%± 4.15%	94.00%± 3.00%	95.50.14%± 3.50%
One Layer (SVC)	79.50%± 10.59%	81.50%± 8.38%	**84.50**%± 8.79%	78.50%± 8.38%	82.50%± 9.29%	78.50%± 7.43%	76.00%± 13.38%
Two Layers (SVC)	**98.0**% ± 3.31%	96.50%± 3.90%	97.50%± 4.03%	**98.00**%± 3.32%	97.00%± 3.32%	96.50%± 3.20%	97.50%± 4.03%

Results obtained with $N^{[1]} = 32$ clusters for layer 1 and $N^{[2]} = 64$ for layer 2 over 10 runs and with pulse characteristics from Table. 1,2 [Main text]

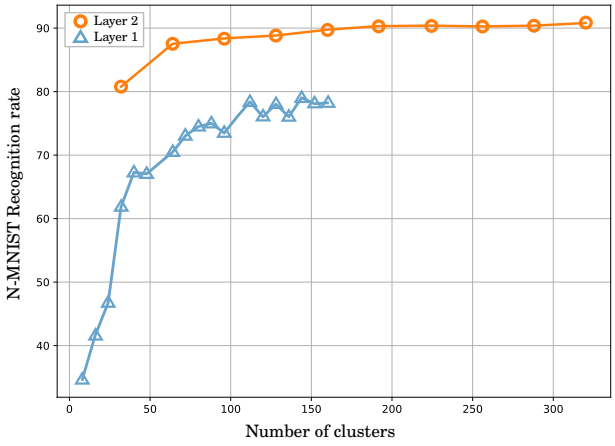

Figure 12: In this figure we show the results N-MNIST accuracy by number of clusters, for a single layer (circles), and a second layer architecture (triangles). Layer 2 results were obtained using 32 clusters in layer 1. As recognition rate quickly reaches an asymptote around 90% recognition rate we decided that an architecture with 32 clusters for layer 1 and 64 clusters for layer 2 was a good compromise between performance and computation time.

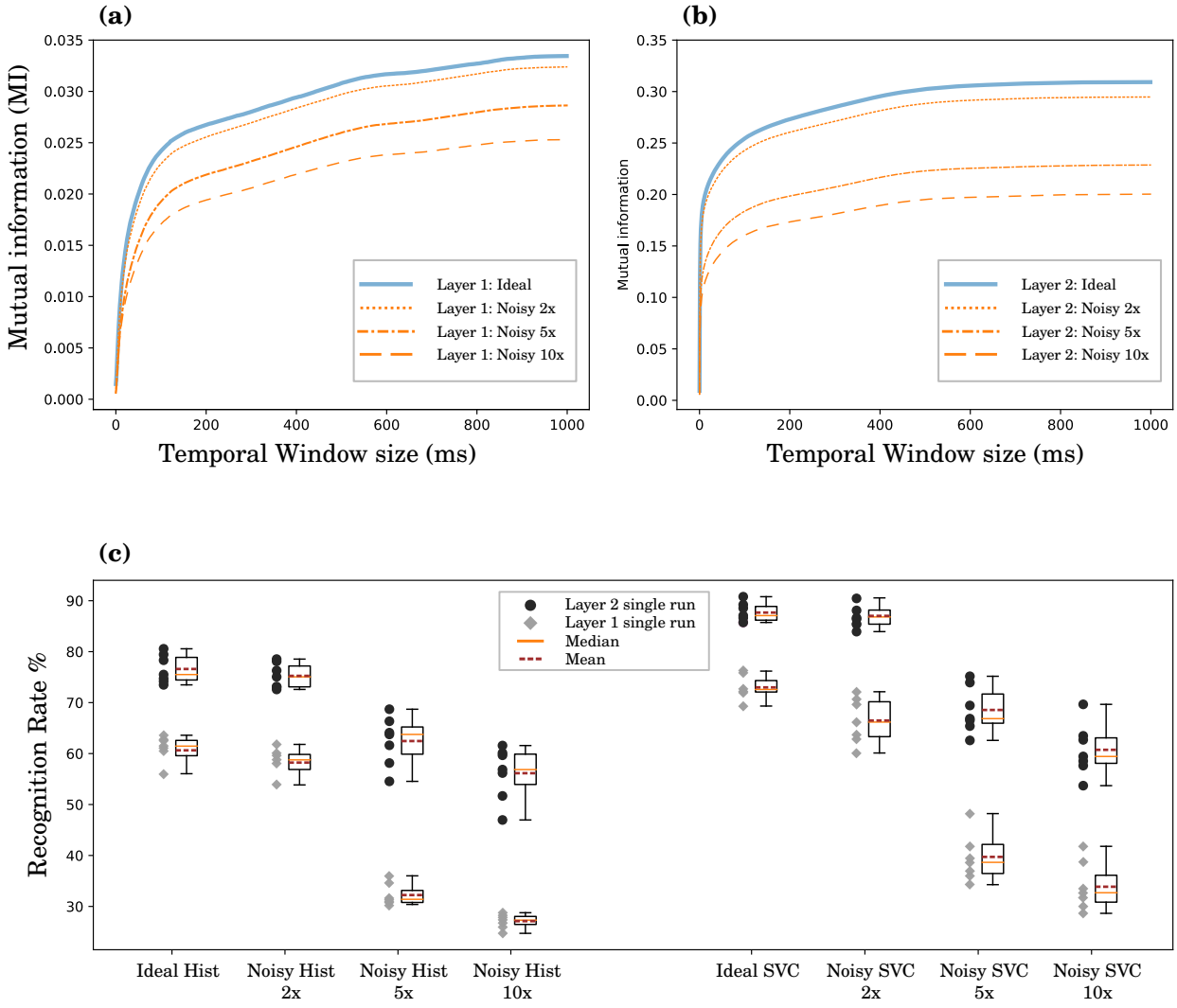

Figure 13: Mutual Information and recognition rates for hypothetical memristors stimulated at 1v 200us but with 2x, 5x, 10x the standard deviations of the stochastic parameters. The plots show that higher degrees of stochasticity can negatively impact Mutual Information of the network (a-b) and Recognition rates (c) for both the tested classifiers (SVC and Hist).

Table 11: Classification accuracy using memristors dynamics vs original exponential decay kernel of HOTS

	Avg. Accuracy ±Std. Deviation	p-value (t-test vs Memristor)	p-value (t-test vs Memristor w/o STP)
Memristor	$61.82\% \pm 1.38\%$	N.A.	$9.26 * 10^{-5}$
Memristor w/o STP	$60.14\% \pm 1.65\%$	$9.26 * 10^{-5}$	N.A.
HOTS Long Decay	$58.93\% \pm 1.48\%$	$2.13 * 10^{-10}$	0.0047
HOTS Short Decay	$55.22\% \pm 1.27\%$	$1.78 * 10^{-26}$	$1.97 * 10^{-18}$

Results obtained with $N^{[1]} = 32$ clusters over 30 consecutive runs.

Bibliography

[1] L. Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Research Bulletin*, 50:303–304, 1999.

[2] A Bolu Ajiboye, Francis R Willett, Daniel R Young, William D Memberg, Brian A Murphy, Jonathan P Miller, Benjamin L Walter, Jennifer A Sweet, Harry A Hoyen, Michael W Keith, et al. Restoration of reaching and grasping movements through brain-controlled muscle stimulation in a person with tetraplegia: a proof-of-concept demonstration. *The Lancet*, 389(10081):1821–1830, 2017.

[3] Himanshu Akolkar, Cedric Meyer, Xavier Clady, Olivier Marre, Chiara Bartolozzi, Stefano Panzeri, and Ryad Benosman. What can neuromorphic event-driven precise timing add to spike-based pattern recognition? *Neural computation*, 27(3):561–593, 2015.

[4] S Ambrogio, S Balatti, F Nardi, S Facchinetti, and D Ielmini. Spike-timing dependent plasticity in a transistor-selected resistive switching memory. *Nanotechnology*, 24(38):384012, 9 2013.

[5] Stefano Ambrogio, Nicola Ciocchini, Mario Laudato, Valerio Milo, Agostino Pirovano, Paolo Fantini, and Daniele Ielmini. Unsupervised learning by spike timing dependent plasticity in phase change memory (pcm) synapses. *Frontiers in neuroscience*, 10:56, 2016.

[6] Haroon Anwar, Xinping Li, Dirk Bucher, and Farzan Nadim. Functional roles of short-term synaptic plasticity with an emphasis on inhibition. *Current Opinion in Neurobiology*, 43:71–78, 4 2017.

[7] Radu Berdan, Eleni Vasilaki, Ali Khiat, Giacomo Indiveri, Alexandru Serb, and Themistoklis Prodromakis. Emulating short-term synaptic dynamics with memristive devices. *Scientific reports*, 6:1–9, 2016.

[8] Fabio Boi, Timoleon Moraitis, Vito De Feo, Francesco Diotalevi, Chiara Bartolozzi, Giacomo Indiveri, and Alessandro Vato. A bidirectional brain-machine interface featuring a neuromorphic hardware decoder. *Frontiers in neuroscience*, 10:563, 2016.

[9] Leon Bottou and Yoshua Bengio. Convergence properties of the k-means algorithms. *Advances in neural information processing systems*, 7, 1994.

[10] Chad E Bouton, Ammar Shaikhouni, Nicholas V Annetta, Marcia A Bockbrader, David A Friedenberg, Dylan M Nielson, Gaurav Sharma, Per B Sederberg, Bradley C Glenn, W Jerry Mysiw, et al. Restoring cortical control of functional movement in a human with quadriplegia. *Nature*, 533(7602):247–250, 2016.

[11] David M Brandman, Sydney S Cash, and Leigh R Hochberg. human intracortical recording and neural decoding for brain–computer interfaces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(10):1687–1696, 2017.

[12] Frédéric D Broccard, Siddharth Joshi, Jun Wang, and Gert Cauwenberghs. Neuromorphic neural interfaces: from neurophysiological inspiration to biohybrid coupling with nervous systems. *Journal of neural engineering*, 14(4):041002, 2017.

[13] Tiago P. Carvalho. A novel learning rule for long-term plasticity of short-term synaptic plasticity enhances temporal processing. *Frontiers in Integrative Neuroscience*, 5 2011.

[14] Peng Chen, Fenghao Liu, Peng Lin, Peihong Li, Yu Xiao, Bihua Zhang, and Gang Pan. Open-loop analog programmable electrochemical memory array. *Nature Communications*, 14(1):6184, 2023.

[15] Elisabetta Chicca, Fabio Stefanini, Chiara Bartolozzi, and Giacomo Indiveri. Neuromorphic electronic circuits for building autonomous cognitive systems. *Proceedings of the IEEE*, 102(9):1367–1388, 2014.

[16] R. Chiu, D. López-Mancilla, Carlos E. Castañeda, Onofre Orozco-López, E. Villafaña-Rauda, and R. Sevilla-Escoboza. Design and implementation of a jerk circuit using a hybrid analog–digital system. *Chaos, Solitons and Fractals*, 119:255–262, 2019.

[17] Adam Coates and Andrew Y. Ng. *Learning Feature Representations with K-Means*, pages 561–580. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[18] Jennifer L Collinger, Brian Wodlinger, John E Downey, Wei Wang, Elizabeth C Tyler-Kabara, Douglas J Weber, Angus JC McMorland, Meel Velliste, Michael L Boninger, and Andrew B Schwartz. High-performance neuroprosthetic control by an individual with tetraplegia. *The Lancet*, 381(9866):557–564, 2013.

[19] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[20] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

[21] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[22] Alan D Degenhart, William E Bishop, Emily R Oby, Elizabeth C Tyler-Kabara, Steven M Chase, Aaron P Batista, and Byron M Yu. Stabilization of a brain–computer interface via the alignment of low-dimensional spaces of neural activity. *Nature biomedical engineering*, 4(7):672–685, 2020.

[23] Yiğit Demirağ, Filippo Moro, Thomas Dalgaty, Gabriele Navarro, Charlotte Frenkel, Giacomo Indiveri, Elisa Vianello, and Melika Payvand. Pcm-trace: scalable synaptic eligibility traces with resistivity drift of phase-change materials. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2021.

[24] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[25] Julie Dethier, Paul Nuyujukian, Stephen I Ryu, Krishna V Shenoy, and Kwabena Boahen. Design and validation of a real-time spiking-neural-network decoder for brain–machine interfaces. *Journal of neural engineering*, 10(3):036008, 2013.

[26] Sukru B. Eryilmaz, Duygu Kuzum, Rakesh Jeyasingh, SangBum Kim, Matthew BrightSky, Chung Lam, and H.-S. Philip Wong. Brain-like associative learning using a nanoscale non-volatile phase change synaptic device array. *Frontiers in Neuroscience*, 8:205, 2014.

[27] Chaofei Fan, Nick Hahn, Foram Kamdar, Donald Avansino, Guy Wilson, Leigh Hochberg, Krishna V Shenoy, Jaimie Henderson, and Francis Willett. Plug-and-play stability for intracortical brain-computer interfaces: A one-year demonstration of seamless brain-to-text communication. *Advances in Neural Information Processing Systems*, 36, 2024.

[28] Daniel B. Fasnacht, Adrian M. Whatley, and Giacomo Indiveri. A serial communication infrastructure for multi-chip address event systems. In *2008 IEEE International Symposium on Circuits and Systems*, pages 648–651, 2008.

[29] Elliot J. Fuller, Farid El Gabaly, François Léonard, Sapan Agarwal, Steven J. Plimpton, Robin B. Jacobs-Gedrim, Conrad D. James, Matthew J. Marinella, and A. Alec Talin. Li-ion synaptic transistor for low power analog computing. *Advanced Materials*, 29, 1 2017.

[30] Elliot J Fuller, Scott T Keene, Armantas Melianas, Zhongrui Wang, Sapan Agarwal, Yiyang Li, Yaakov Tuchman, Conrad D James, Matthew J Marinella, J Joshua Yang, et al. Parallel programming of an ionic floating-gate memory array for scalable neuromorphic computing. *Science*, 364(6440):570–574, 2019.

[31] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(1):154–180, 2020.

[32] Gregory Gauvain, Himanshu Akolkar, Antoine Chaffiol, Fabrice Arcizet, Mina A Khoei, Mélissa Desrosiers, Céline Jaillard, Romain Caplette, Olivier Marre, Stéphane Bertin, et al. Optogenetic therapy: high spatiotemporal resolution and pattern discrimination compatible with vision restoration in non-human primates. *Communications biology*, 4(1):1–15, 2021.

[33] Daniel Gehrig, Antonio Loquercio, Konstantinos G Derpanis, and Davide Scaramuzza. End-to-end learning of representations for asynchronous event-based data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5633–5643, 2019.

[34] Antoine Grimaldi, Victor Boutin, Sio-Hoi Ieng, Ryad Benosman, and Laurent Perrinet. A robust event-driven approach to always-on object recognition. *Authorea Preprints*, 2023.

[35] Antoine Grimaldi, Victor Boutin, Laurent Perrinet, Sio-Hoi Ieng, and Ryad Benosman. A homeostatic gain control mechanism to improve event-driven object recognition. In *2021 International Conference on Content-Based Multimedia Indexing (CBMI)*, pages 1–6, 2021.

[36] Marina Gul and M Abdul Rehman. Big data: an optimized approach for cluster initialization. *Journal of Big Data*, 10(1):120, 2023.

[37] Muhammad Haris, Greg Shakhnarovich, and Norimichi Ukita. Space-time-aware multi-resolution video enhancement. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2856–2865, 2020.

[38] Andreas V. M. Herz, Tim Gollisch, Christian K. Machens, and Dieter Jaeger. Modeling single-neuron dynamics and computations: A balance of detail and abstraction. *Science*, 314(5796):80–85, 2006.

[39] Leigh R Hochberg, Daniel Bacher, Beata Jarosiewicz, Nicolas Y Masse, John D Simeral, Joern Vogel, Sami Haddadin, Jie Liu, Sydney S Cash, Patrick Van Der Smagt, et al. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485(7398):372–375, 2012.

[40] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[41] Qilin Hua, Huaqiang Wu, Bin Gao, Qingtian Zhang, Wei Wu, Yujia Li, Xiaohu Wang, Weiguo Hu, and He Qian. Low-voltage oscillatory neurons for memristor-based neuromorphic systems. *Global Challenges*, 3:1900015, 11 2019.

[42] Sio-Hoi Ieng, Joao Carneiro, Marc Osswald, and Ryad Benosman. Neuromorphic Event-Based Generalized Time-Based Stereovision. *Frontiers in Neuroscience*, 12:442, 2018.

[43] IGN. Gabe newell talks half-life: Alyx and valve's past and (unexpected) future – ign first, 2020.

[44] Giacomo Indiveri. Modeling Selective Attention Using a Neuromorphic Analog VLSI Device. *Neural Computation*, 12(12):2857–2880, 12 2000.

[45] Giacomo Indiveri, Federico Corradi, and Ning Qiao. Neuromorphic architectures for spiking deep neural networks. In *2015 IEEE International Electron Devices Meeting (IEDM)*, pages 4.2.1–4.2.4, 2015.

[46] Laxmi R. Iyer, Yansong Chua, and Haizhou Li. Is neuromorphic mnist neuromorphic? analyzing the discriminative power of neuromorphic datasets in the time domain. *Frontiers in Neuroscience*, 15, 2021.

[47] Adam Jaeger and David Banks. Cluster analysis: A modern statistical review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 15(3):e1597, 2023.

[48] YeonJoo Jeong, Jihang Lee, John Moon, Jong Hoon Shin, and Wei D. Lu. K-means data clustering with memristor networks. *Nano Letters*, 18(7):4447–4453, 2018. PMID: 29879355.

[49] Bowen Ji, Zekai Liang, Xichen Yuan, Honglai Xu, Minghao Wang, Erwei Yin, Zhejun Guo, Longchun Wang, Yuhao Zhou, Huicheng Feng, et al. Recent advances in wireless epicortical and intracortical neuronal recording systems. *Science China Information Sciences*, 65(4):1–18, 2022.

[50] D. Kuzum, R. G. D. Jeyasingh, and H. . P. Wong. Energy efficient programming of nanoelectronic synaptic devices for large-scale implementation of associative and temporal sequence learning. In *2011 International Electron Devices Meeting*, pages 30.3.1–30.3.4, 2011.

[51] Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E Shi, and Ryad B Benosman. Hots: a hierarchy of event-based time-surfaces for pattern recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1346–1359, 2016.

[52] Christiaan Lamers, René Vidal, Nabil Belbachir, Niki van Stein, Thomas Bäeck, and Paris Giampouras. Clustering-based domain-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3384–3392, 2023.

[53] Hua-Min Li, Ke Xu, Buchanan Bourdon, Hao Lu, Yu-Chuan Lin, Joshua A Robinson, Alan C Seabaugh, and Susan K Fullerton-Shirey. Electric double layer dynamics in poly (ethylene oxide) liclo4 on graphene transistors. *The Journal of Physical Chemistry C*, 121(31):16996–17004, 2017.

[54] Yiyang Li. A review of recent research on nonequilibrium solid solution behavior in lixfepo4. *Solid State Ionics*, 323:142–150, 10 2018.

[55] Yiyang Li, Elliot J Fuller, Shiva Asapu, Sapan Agarwal, Tomochika Kurita, J Joshua Yang, and A Alec Talin. Low-voltage, cmos-free synaptic memory based on lix tio2 redox transistors. *ACS applied materials & interfaces*, 11(42):38982–38992, 2019.

[56] Yiyang Li, Elliot J Fuller, Joshua D Sugar, Sangmin Yoo, David S Ashby, Christopher H Bennett, Robert D Horton, Michael S Bartsch, Matthew J Marinella, Wei D Lu, et al. Filament-free bulk resistive memory enables deterministic analogue switching. *Advanced Materials*, 32(45):2003984, 2020.

[57] Yue Li, Zihao Xuan, Jikai Lu, Zhongrui Wang, Xumeng Zhang, Zuheng Wu, Yongzhou Wang, Han Xu, Chunmeng Dou, Yi Kang, et al. One transistor one electrolyte-gated transistor based spiking neural network for power-efficient neuromorphic computing system. *Advanced Functional Materials*, page 2100042, 2021.

[58] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128 x 128 120db 30mw asynchronous vision sensor that responds to relative intensity change. In *2006 IEEE International Solid State Circuits Conference-Digest of Technical Papers*, pages 2060–2069. IEEE, 2006.

[59] Hyungkwang Lim, Hyung-Woo Ahn, Vladimir Kornijcuk, Guhyun Kim, Jun Yeong Seok, Inho Kim, Cheol Seong Hwang, and Doo Seok Jeong. Relaxation oscillator-realized artificial electronic neurons, their responses, and noise. *Nanoscale*, 8:9629–9640, 2016.

[60] Carver Mead and Ismail Mohammed. *Analog VLSI Implementation of Neural Systems*, volume 80. Springer New York, NY, 1989.

[61] Yuanyuan Mi, Xiaohan Lin, and Si Wu. Neural computations in a dynamical system with multiple time scales. *Frontiers in Computational Neuroscience*, 10, 9 2016.

[62] Saber Moradi, Ning Qiao, Fabio Stefanini, and Giacomo Indiveri. A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). *IEEE transactions on biomedical circuits and systems*, 12(1):106–122, 2017.

[63] David A Moses, Sean L Metzger, Jessie R Liu, Gopala K Anumanchipalli, Joseph G Makin, Pengfei F Sun, Josh Chartier, Maximilian E Dougherty, Patricia M Liu, Gary M Abrams, et al. Neuroprosthesis for decoding speech in a paralyzed person with anarthria. *New England Journal of Medicine*, 385(3):217–227, 2021.

[64] Elon Musk et al. An integrated brain-machine interface platform with thousands of channels. *Journal of medical Internet research*, 21(10):e16194, 2019.

[65] Manu V Nair and Giacomo Indiveri. An ultra-low power sigma-delta neuron circuit. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2019.

[66] Arto Nurmikko. Challenges for large-scale cortical interfaces. *Neuron*, 108(2):259–269, 2020.

[67] Takeo Ohno, Tsuyoshi Hasegawa, Tohru Tsuruoka, Kazuya Terabe, James K Gimzewski, and Masakazu Aono. Short-term plasticity and long-term potentiation mimicked in single inorganic synapses. *Nature materials*, 10:591–595, 2011.

[68] Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 9:437, 2015.

[69] Jelili Oyelade, Itunuoluwa Isewon, Funke Oladipupo, Olufemi Aromolaran, Efosa Uwoghiren, Faridah Ameh, Moses Achas, and Ezekiel Adebiyi. Clustering algorithms: their application to gene expression data. *Bioinformatics and Biology insights*, 10:BBI–S38316, 2016.

[70] Eustace Painkras, Luis A Plana, Jim Garside, Steve Temple, Francesco Galluppi, Cameron Patterson, David R Lester, Andrew D Brown, and Steve B Furber. Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation. *IEEE Journal of Solid-State Circuits*, 48(8):1943–1953, 2013.

[71] Jongkil Park, Theodore Yu, Siddharth Joshi, Christoph Maier, and Gert Cauwenberghs. Hierarchical address event routing for reconfigurable large-scale neuromorphic systems. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2408–2422, 2017.

[72] János A Perge, Mark L Homer, Wasim Q Malik, Sydney Cash, Emad Eskandar, Gerhard Friehs, John P Donoghue, and Leigh R Hochberg. Intra-day signal instabilities affect decoding performance in an intracortical neural interface system. *Journal of neural engineering*, 10(3):036004, 2013.

[73] Matthew D. Pickett, Gilberto Medeiros-Ribeiro, and R. Stanley Williams. A scalable neuristor built with mott memristors. *Nature Materials*, 12:114–117, 2 2013.

[74] Harry A. Pierson and Michael S. Gashler. Deep learning in robotics: a review of recent research. *Advanced Robotics*, 31(16):821–835, 2017.

[75] Mahardhika Pratama, Andri Ashfahani, and Edwin Lughofer. Unsupervised continual learning via self-adaptive deep clustering approach. In *International Workshop on Continual Semi-Supervised Learning*, pages 48–61. Springer, 2021.

[76] Dale Purves, G Augustine, D Fitzpatrick, L Katz, A LaMantia, J McNamara, and S Williams. Neuroscience 2nd edition. In *Neuroscience 2nd edition*, chapter 7: Two Families of Postsynaptic Receptors. Sinauer Associates, 2001.

[77] Chuan Qian, Jia Sun, Ling-an Kong, Guangyang Gou, Junliang Yang, Jun He, Yongli Gao, and Qing Wan. Artificial synapses based on in-plane gate organic electrochemical transistors. *ACS applied materials & interfaces*, 8(39):26169–26175, 2016.

[78] Ning Qiao, Hesham Mostafa, Federico Corradi, Marc Osswald, Fabio Stefanini, Dora Sumislawska, and Giacomo Indiveri. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in Neuroscience*, 9:141, 2015.

[79] Adrien B Rapeaux and Timothy G Constandinou. Implantable brain machine interfaces: first-in-human studies, technology challenges and trends. *Current opinion in biotechnology*, 72:102–111, 2021.

[80] Marco Rasetto and Himanshu Akolkar. Sup3r: A semi-supervised algorithm for increasing sparsity, stability, and separability in hierarchy of time-surfaces architectures. *arXiv preprint arXiv:2404.12402*, 2024.

[81] Marco Rasetto, Juan P Dominguez-Morales, Angel Jimenez-Fernandez, and Ryad Benosman. Event based time-vectors for auditory features extraction: a neuromorphic approach for low power audio recognition. *arXiv preprint arXiv:2112.07011*, 2021.

[82] Marco Rasetto, Qingzhou Wan, Himanshu Akolkar, Feng Xiong, Bertram Shi, and Ryad Benosman. Building time-surfaces by exploiting the complex volatility of an ecram memristor. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2023.

[83] Syed Ghazi Sarwat, Benedikt Kersting, Timoleon Moraitis, Vara Prasad Jonnalagadda, and Abu Sebastian. Phase-change memtransistive synapses for mixed-plasticity neural computations. *Nature Nanotechnology*, 17:507–513, 5 2022.

[84] Johannes Schemmel, Johannes Fieres, and Karlheinz Meier. Wafer-scale integration of analog neural networks. In *2008 IEEE International Joint Conference on Neu-*

ral Networks (IEEE World Congress on Computational Intelligence), pages 431–438, 2008.

[85] Catherine D Schuman, Thomas E Potok, Robert M Patton, J Douglas Birdwell, Mark E Dean, Garrett S Rose, and James S Plank. A survey of neuromorphic computing and neural networks in hardware. *arXiv preprint arXiv:1705.06963*, 2017.

[86] Andrew B Schwartz, X Tracy Cui, Douglas J Weber, and Daniel W Moran. Brain-controlled interfaces: movement restoration with neural prosthetics. *Neuron*, 52(1):205–220, 2006.

[87] Andrew B Schwartz, Ronald E Kettner, and Apostolos P Georgopoulos. Primate motor cortex and free arm movements to visual targets in three-dimensional space. i. relations between single cell discharge and direction of movement. *Journal of Neuroscience*, 8(8):2913–2927, 1988.

[88] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178, 2010.

[89] Teresa Serrano-Gotarredona and Bernab'e Linares-Barranco. Poker-dvs and mnist-dvs. their history, how they were made, and other details. *Frontiers in neuroscience*, 9:481, 2015.

[90] Teresa Serrano-Gotarredona and Bernabé Linares-Barranco. A 128×128 1.5% contrast sensitivity 0.9% fpn 3 µs latency 4 mw asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers. *IEEE Journal of Solid-State Circuits*, 48(3):827–838, 2013.

[91] Ahmed Shaban, Sai Sukruth Bezugam, and Manan Suri. An adaptive threshold neuron for recurrent spiking neural networks with nanodevice hardware implementation. *Nature Communications*, 12(1):4234, 2021.

[92] Shoeb Shaikh, Rosa So, Tafadzwa Sibindi, Camilo Libedinsky, and Arindam Basu. Sparse ensemble machine learning to improve robustness of long-term decoding in ibmis. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 28(2):380–389, 2019.

[93] Mohammad Taghi Sharbati, Yanhao Du, Jorge Torres, Nolan D Ardolino, Minhee Yun, and Feng Xiong. Low-power, electrochemically tunable graphene synapses for neuromorphic computing. *Advanced Materials*, 30(36):1802353, 2018.

[94] Mohammadali Sharifshazileh, Karla Burelo, Johannes Sarnthein, and Giacomo Indiveri. An electronic neuromorphic system for real-time detection of high frequency oscillations (hfo) in intracranial eeg. *Nature communications*, 12(1):1–14, 2021.

[95] Sumit B Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *Advances in neural information processing systems*, 31, 2018.

[96] John D Simeral, Thomas Hosman, Jad Saab, Sharlene N Flesher, Marco Vilela, Brian Franco, Jessica N Kelemen, David M Brandman, John G Ciancibello, Paymon G Rezaii, et al. Home use of a percutaneous wireless intracortical brain-computer interface by individuals with tetraplegia. *IEEE Transactions on Biomedical Engineering*, 68(7):2313–2325, 2021.

[97] Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. Hats: Histograms of averaged time surfaces for robust event-based object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1731–1740, 2018.

[98] Sergey D Stavisky, Francis R Willett, Donald T Avansino, Leigh R Hochberg, Krishna V Shenoy, and Jaimie M Henderson. Speech-related dorsal motor cortex activity does not interfere with ibci cursor control. *Journal of neural engineering*, 17(1):016049, 2020.

[99] Sergey D Stavisky, Francis R Willett, Guy H Wilson, Brian A Murphy, Paymon Rezaii, Donald T Avansino, William D Memberg, Jonathan P Miller, Robert F Kirsch, Leigh R Hochberg, et al. Neural ensemble dynamics in dorsal motor cortex during speech in people with paralysis. *Elife*, 8:e46015, 2019.

[100] D. B. Strukov, D. R. Stewart, J. Borghetti, X. Li, M. Pickett, G. Medeiros Ribeiro, W. Robinett, G. Snider, J. P. Strachan, W. Wu, Q. Xia, J. Joshua Yang, and R. S. Williams. Hybrid cmos/memristor circuits. In *2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1967–1970, 2010.

[101] Dmitri Strukov and Stan Williams. Exponential ionic drift: Fast switching and low volatility of thin-film memristors. *Applied Physics A*, 94:515–519, 03 2009.

[102] Linfeng Sun, Yishu Zhang, Geunwoo Hwang, Jinbao Jiang, Dohyun Kim, Yonas Assefa Eshete, Rong Zhao, and Heejun Yang. Synaptic computation enabled by joule heating of single-layered semiconductors for sound localization. *Nano Letters*, 18:3229–3234, 2018.

[103] Sang Hyun Sung, Tae Jin Kim, Hyera Shin, Tae Hong Im, and Keon Jae Lee. Simultaneous emulation of synaptic and intrinsic plasticity using a memristive synapse. *Nature Communications*, 13:2811, 12 2022.

[104] Manan Suri, Olivier Bichler, Damien Querlioz, Olga Cueto, Luca Perniola, Veronique Sousa, Dominique Vuillaume, Christian Gamrat, and Barbara DeSalvo. Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction. In *2011 International Electron Devices Meeting*, pages 4–4. IEEE, 2011.

[105] Jianshi Tang, Douglas Bishop, Seyoung Kim, Matt Copel, Tayfun Gokmen, Teodor Todorov, SangHoon Shin, Ko-Tao Lee, Paul Solomon, Kevin Chan, et al. Ecram as scalable synaptic cell for high-speed, low-power neuromorphic computing. In *2018 IEEE International Electron Devices Meeting (IEDM)*, pages 13–1. IEEE, 2018.

[106] Ricardo Tapiador-Morales, Jean-Matthieu Maro, Angel Jimenez-Fernandez, Gabriel Jimenez-Moreno, Ryad Benosman, and Alejandro Linares-Barranco. Event-based gesture recognition through a hierarchy of time-surfaces for fpga. *Sensors*, 20(12):3404, 2020.

[107] Yoeri van de Burgt, Ewout Lubberman, Elliot J Fuller, Scott T Keene, Grégorio C Faria, Sapan Agarwal, Matthew J Marinella, A Alec Talin, and Alberto Salleo. A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing. *Nature materials*, 16(4):414–418, 2017.

[108] A. F. Vincent, J. Larroque, N. Locatelli, N. Ben Romdhane, O. Bichler, C. Gamrat, W. S. Zhao, J. Klein, S. Galdin-Retailleau, and D. Querlioz. Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems. *IEEE Transactions on Biomedical Circuits and Systems*, 9(2):166–174, 2015.

[109] Qingzhou Wan, Marco Rasetto, Mohammad T Sharbati, John R Erickson, Sridhar Reddy Velagala, Matthew T Reilly, Yiyang Li, Ryad Benosman, and Feng Xiong. Low-voltage electrochemical lixwo3 synapses with temporal dynamics for spiking neural networks. *Advanced Intelligent Systems*, 3(9):2100021, 2021.

[110] Qingzhou Wan, Marco Rasetto, Mohammad T Sharbati, John R Erickson, Sridhar Reddy Velagala, Matthew T Reilly, Yiyang Li, Ryad Benosman, and Feng Xiong. Low-voltage electrochemical lixwo3 synapses with temporal dynamics for spiking neural networks. *Advanced Intelligent Systems*, n/a:2100021, 2021.

[111] Qingzhou Wan, Mohammad T. Sharbati, John R. Erickson, Yanhao Du, and Feng Xiong. Emerging artificial synaptic devices for neuromorphic computing. *Advanced Materials Technologies*, 4:1900037, 4 2019.

[112] Qingzhou Wan, Peng Zhang, Qiming Shao, Mohammad T. Sharbati, John R. Erickson, Kang L. Wang, and Feng Xiong. (bi0.2sb0.8)2te3 based dynamic synapses with programmable spatio-temporal dynamics. *APL Materials*, 7, 10 2019.

[113] Po-Min Wang, Stanislav Culaclii, Kyung Jin Seo, Yushan Wang, Hui Fang, Yi-Kai Lo, and Wentai Liu. Challenges in the design of large-scale, high-density, wireless stimulation and recording interface. *Interfacing Bioelectronics and Biomedical Sensing*, pages 1–28, 2020.

[114] Zhongrui Wang, Saumil Joshi, Sergey Savel'ev, Wenhao Song, Rivu Midya, Yunning Li, Mingyi Rao, Peng Yan, Shiva Asapu, Ye Zhuo, Hao Jiang, Peng Lin, Can Li, Jung Ho Yoon, Navnidhi K. Upadhyay, Jiaming Zhang, Miao Hu, John Paul Strachan, Mark Barnell, Qing Wu, Huaqiang Wu, R. Stanley Williams, Qiangfei Xia, and J. Joshua Yang. Fully memristive neural networks for pattern classification with unsupervised learning. *Nature Electronics*, 1:137–145, 2 2018.

[115] Francis R Willett, Donald T Avansino, Leigh R Hochberg, Jaimie M Henderson, and Krishna V Shenoy. High-performance brain-to-text communication via handwriting. *Nature*, 593(7858):249–254, 2021.

[116] Graham Williams and David C Watts. Non-symmetrical dielectric relaxation behaviour arising from a simple empirical decay function. *Transactions of the Faraday society*, 66:80–85, 1970.

[117] B Wodlinger, JE Downey, EC Tyler-Kabara, AB Schwartz, ML Boninger, and JL Collinger. Ten-dimensional anthropomorphic arm control in a human brain- machine interface: difficulties, solutions, and limitations. *Journal of neural engineering*, 12(1):016011, 2014.

[118] Jonathan R. Wolpaw. Chapter 6 - brain–computer interfaces. In Michael P. Barnes and David C. Good, editors, *Neurological Rehabilitation*, volume 110 of *Handbook of Clinical Neurology*, pages 67–74. Elsevier, 2013.

[119] Qiangfei Xia and J. Joshua Yang. Memristive crossbar arrays for brain-inspired computing. *Nature Materials*, 18:309–323, 4 2019.

[120] Chuan-Sen Yang, Da-Shan Shang, Nan Liu, Elliot J Fuller, Sapan Agrawal, A Alec Talin, Yong-Qing Li, Bao-Gen Shen, and Young Sun. All-solid-state synaptic transistor with ultralow conductance for neuromorphic computing. *Advanced Functional Materials*, 28(42):1804170, 2018.

[121] Rui Yang, Kazuya Terabe, Yiping Yao, Tohru Tsuruoka, Tsuyoshi Hasegawa, James K Gimzewski, and Masakazu Aono. Synaptic plasticity and memory functions achieved in a wo3- x-based nanoionics device by using the principle of atomic switch operation. *Nanotechnology*, 24:384003, 2013.

[122] Xiahui Yao, Konstantin Klyukin, Wenjie Lu, Murat Onen, Seungchan Ryu, Dongha Kim, Nicolas Emond, Iradwikanari Waluyo, Adrian Hunt, Jesús A Del Alamo, et al. Protonic solid-state electrochemical synapse for physical neural networks. *Nature communications*, 11(1):1–10, 2020.

[123] BM Yu, G Santhanam, M Sahani, and KV Shenoy. Chapter 7-neural decoding for motor and communication prostheses, 2010.

[124] Zhiting Zhang and Ji Dai. Fully implantable wireless brain-computer interface for humans: Advancing toward the future. *The Innovation*, 2024.

[125] Jiadi Zhu, Yuchao Yang, Rundong Jia, Zhongxin Liang, Wen Zhu, Zia Ur Rehman, Lin Bao, Xiaoxian Zhang, Yimao Cai, Li Song, et al. Ion gated synaptic transistors based on 2d van der waals crystals with tunable diffusive dynamics. *Advanced Materials*, 30(21):1800195, 2018.