A robust event-driven approach to always-on object recognition

Antoine Grimaldi[1], Victor Boutin[1], Sio-Hoi Ieng[2], Ryad Benosman[2] and Laurent U Perrinet[1]

Abstract— We propose a neuromimetic architecture that can perform always-on pattern recognition. To achieve this, we extended an existing event-based algorithm [1], which introduced novel spatio-temporal features as a Hierarchy Of Time-Surfaces (HOTS). Built from asynchronous events acquired by a neuromorphic camera, these time surfaces allow to code the local dynamics of a visual scene and to create an efficient event-based pattern recognition architecture. Inspired by neuroscience, we extended this method to increase its performance. First, we add a homeostatic gain control on the activity of neurons to improve the learning of spatio-temporal patterns [2]. We also provide a new mathematical formalism that allows to draw an analogy between the HOTS algorithm and Spiking Neural Networks (SNN). Following that analogy, we remodel the offline pattern categorization method previously used into an online and event-driven layer. This classifier uses the spiking output of the network to define novel time surfaces and we then perform online classification with a neuromimetic implementation of a multinomial logistic regression. Not only do these improvements increase consistently the performances of the network, they also make this event-driven pattern recognition algorithm online. Results were validated on different datasets: Poker-DVS [3], N-MNIST [4] and DVS Gesture [5]. This demonstrates the efficiency of this bio-realistic SNN for ultra-fast object categorization through an event-by-event decision making process.

Index Terms—vision, pattern recognition, event-based computations, spiking neural networks, homeostasis, efficient coding, online classification

1 Introduction

Bio-inspired engineering aims at taking advantage of our understanding of the complex and impressively efficient mechanisms found in nature. Event-based cameras perfectly illustrate this process. Also called silicon retinas, these sensors are inspired by biological retinas and make it possible to capture luminous information asynchronously. Unlike its classical frame-based counterpart, an event-based camera responds to the scene's dynamics in a pixel-wise fashion: when a change in luminance is detected, an event is emitted. The event is labeled with an ON or OFF polarity whether it corresponds to an increase or decrease in brightness, respectively (see figure 1). Event-based cameras offer various advantages and notably a high temporal resolution, energy efficiency, redundancy reduction, and a high dynamic range. Numerous interesting applications and use cases of event-based cameras are nowadays flourishing in the scientific community (see [6] for a review). This new technology, along with the corresponding Address Event Representation specification [7], brings a paradigm shift in the way visual information is processed. Efficient event-driven solutions were found to solve classical computer vision tasks such as estimating optical flow [8–10], inferring 3D reconstruction [11–13] or solving the simultaneous localization and mapping problem [14, 15]. In this work, we focus on performing pattern recognition and extend an already existing method [1].

This particular model performs object recognition thanks to a feedforward hierarchical architecture using *time surfaces*, an event-driven analog representation of the local dynamics of a scene. Then, these are assembled in a Hierarchy Of Time Surfaces (HOTS). Using a form of Hebbian learning, the network is able to learn, in an unsupervised way, progressively more complex spatio-temporal features which appear in the event stream. This algorithm was shown to make accurate predictions on a letter and digit dataset [16], on a flipped card dataset [17] and on a dataset of scenes with faces.

We identified two main limitations in the HOTS algorithm. First, the unsupervised clustering of the kernels depends highly on initialization, and this may impact the performance of the network. We have recently proposed to include a bio-plausible homeostatic gain control mechanism [2]. We showed there that unsupervised learning of the features is qualitatively improved by balancing the activity of the different neurons within the same layer. We moreover tested the classification accuracy over different datasets by injecting different amounts of spatial and temporal noise to the stream of input events, proving that efficiency was increased by homeostasis. The second limitation that we identified in the HOTS method is the output classifier. Indeed, it is based on the computation of a histogram of neural activations in the final layer of the network to

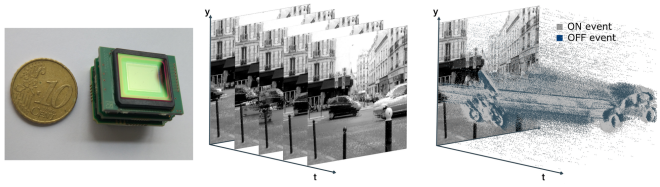

Fig. 1: A miniature event-based ATIS sensor (Left) which, compared to classical frame-based representations (Middle), outputs an event-based representation of the scene (Right).

perform the classification. Such a method discounts the fine-grained temporal dynamics of the stream of events emitted by the last layer of the network. More importantly, a major drawback is that classification can only be performed *post hoc*, once all the events of the tested sample were received.

This offline layer was chosen as it gave an accurate response once all events were emitted. In this study, we include and test an online classification method at the last layer of the network, such that we achieve a fully end-to-end event-driven and online pattern categorization. We formally demonstrate that the whole structure of the proposed model corresponds to a biologically plausible Spiking Neural Network (SNN). To our knowledge, it is the first always-on, event-by-event object recognition method, and we validate it on different datasets designed for symbol [3], digit [4] or gesture [5] categorization. Given the simplicity of the architecture of the proposed network, its event-based formalism and its local learning rules, this method is easily transferable to neuromorphic hardware to use the efficiency of event-based computing.

In that perspective, this paper is organized as follows. First, we present the HOTS algorithm using a novel mathematical formalization and the improvements brought by our method. Then, we extend the categorization algorithm by including a simple bio-plausible online classification layer. We prove that our method corresponds to a SNN with Leaky Integrate-and-Fire (LIF) neuron models that can be implemented in a neuromorphic chip. Finally, we show the quantitative improvements of the resulting classification performance, and how its dynamics may vary for different datasets. We have tested the model for different event-camera datasets, and a full implementation of this algorithm is available at https://github.com/AntoineGrimaldi/hotsline. These scripts allow reproducing all results presented in this paper, and we give links to reproducible notebooks within the text.

2 Materials and methods

In this section, we start by describing the datasets used in this study and presenting the method we designed to test our algorithm's robustness to spatial and temporal jitter. After providing an overview of existing object recognition algorithms, we generalize the event-based HOTS model, already described in [1], and extend its formalism to the continuous time domain. Then, we present the improvements making the algorithm fully online and bio-plausible. We introduce the homeostasis regulation rule that allows for a better learning of the weights of the different layers [2], and we describe a new classifier using Multinomial Logistic Regression (MLR) to propose an end-to-end event-driven classification algorithm. We close this section by providing a formal analogy of our architecture with a SNN and local correlation-based learning rules to propose a unified theoretical framework between neuromorphic engineering and computational neuroscience.

2.1 Datasets

To load the events, we use the community-built *tonic* python package [18]. It currently offers the possibility to load 12 different event-based vision datasets and is based on the PyTorch language [19]. This allows to load event streams in a standard fashion and to optionally apply data augmentation methods to the event streams. Once loaded, an event-based camera recording is a $N_{ev} \times 4$ matrix in which N_{ev} represents the number of events and the 4 columns represent respectively the x and y positions on the pixel grid, the timestamp value, and the polarity. Timestamps are given in microseconds and polarities are 0 and 1 respectively for OFF and ON events. Among these datasets, 6 of them are labeled for object classification tasks. We choose to test the performances of our method on 3 different datasets:

- **Poker-DVS dataset** [3], one of the first publicly available DVS recordings from a real-world scene that was used to test performances of HOTS [1]. It consists of 131 occurrences of the four different symbols of playing poker cards (clubs, diamonds, hearts and spades).
- **N-MNIST dataset** [4], a widely used dataset that was recorded while moving an event-based camera in front of a screen on which digitized MNIST digits [20] were projected.
- **DVS128 Gesture dataset** [5], that is composed of more complex and naturalistic recordings on real-world scenes. In that dataset, 29 subjects perform hands and arms gestures of different categories ("hand clap", "arm roll", ...). These were recorded with an iniLabs DVS128 event-based camera (with a resolution of 128×128 pixels) and under different lighting conditions. Samples provided by the dataset are 6 seconds long and, to reduce the computational load of the training, we keep only the first second of recording.

To test for the robustness of the proposed algorithm, we also used the *tonic* package to transform and augment the datasets. In particular, this allows to add spatial or temporal jitter to the input stream. As relevant information is supposed to be represented within the timing and position of input events, we can assume that classification performance should get worse as the jitter increases. Therefore, we will use this module to test the robustness of the algorithm by progressively adding some noise to the input signal. To account for the variability of the random jitter applied, we repeat the prediction 10 times for each amount of jitter and derive a statistical quantification from these repetitions. To reduce the simulation time, we compute this study on Poker-DVS, on a subset of N-MNIST and do not perform this analysis on DVSGesture. The subset of N-MNIST is composed of 1000 randomly selected digits with a balanced number of samples between each class.

2.2 Related work

Poker-DVS [3] is included as it was tested with the original HOTS method [1]. Because of its small size, this experiment acts as a toy model to test the different methods. In this work, we focus on performing pattern recognition. A widely used dataset designed for such a task is the event-based, augmented version of MNIST called N-MNIST [4]. Some related approaches have used standard artificial neural networks which were converted to SNN, resulting in overall

good classification results [21, 22]. In 2020, [22] is the first neuromorphic hardware implementation of the event-based N-MNIST benchmark. Alternatively, some other competitive event-driven algorithms are developed using Back-Propagation (BP) adapted for SNN [23–25]. More recently, it was proposed to introduce biomimetic saccades to boost object recognition [26]. All these contributions saturate the digit recognition problem introduced by the N-MNIST dataset with accuracies around 99% but none of them addressed the question of ultra-fast object recognition, i.e. the ability to recognize a digit with only the first events. We report that online inference on event-based data was developed in past studies [27–29]. However, [29] accumulates spikes as input to reconstruct an image frame, [28] uses a sample-and-hold approach, and freezes events during a defined time step. [27] proposes to use Spike-Timing Dependent Plasticity (STDP) for unsupervised learning of spatio-temporal features. For this last study, they also construct a supervised classifier able to learn in an online fashion and that should be able to make an inference for every event. However, they perform a classification based on the strongest response of one neuron during the time window in which the sample is presented. This way, they do not make use of the event-driven nature of the input to the classification layer.

Then, the DVS128 Gesture dataset [5] offers a more complex recognition task as it is composed of real-world scenes with natural movements performed by subjects. [30] presents a non-local synaptic modification method with spiking and artificial neurons inspired from natural networks called self-BP. A spiking version of the deep ResNet architecture (STS-ResNet) achieves good performances for gesture recognition as well [31]. A recent study develops a bio-plausible method using SNN and STDP, reaching very good results [32]. The classifier is a Support Vector Machine (SVM) applied on the feature vectors as output of the SNN. A promising method including Spiking Recurrent Neural Networks (SRNN) is introduced in [33] and obtains high, online performances on several datasets. Accuracy on the DVS128 Gesture dataset reaches 97.61%, yet this method requires an additional preprocessing with a convolution layer on frames computed from the DVS recording and it was not mentioned if this number reflects an average computed on the online accuracy or not. SLAYER [23] could also provide an online classification with a SNN using 8 layers trained with BP. However, the network is trained with a target spike count to distinguish the true class, and requires waiting until the end of the output spike train to infer a decision.

For both datasets, these different methods reach very good performances yet none of them report the accuracy as a function of the number of events which were used, or as a function of time - thus making it impossible to derive online accuracy results. An exception in the literature is [34] that reports a figure of the accuracy as a function of the latency. It is computed on the N-CARS dataset which was created for that study. However, this method uses an accumulation of time surfaces and can not perform an event-by-event classification. The method we propose is the first to develop an *always-on* decision process and we can then report its performances as a function of the number of events integrated by the network. We stick to the study of the three widely used datasets mentioned (N-MNIST, DVSGesture and Poker-DVS) which can be loaded with the *tonic* package.

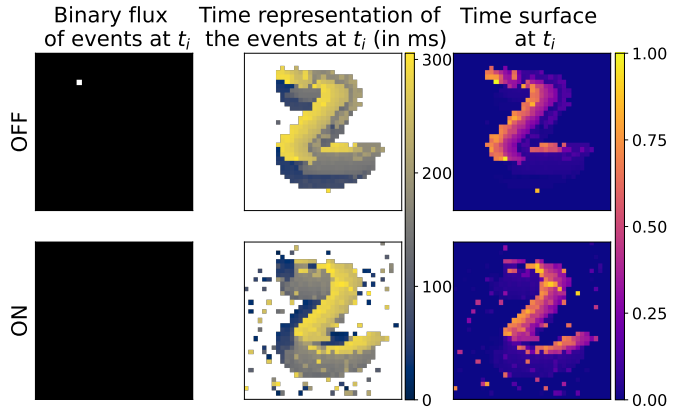

Fig. 2: Illustration of the different event-based data types used in the HOTS network at a given event time. The two rows correspond to the OFF and ON polarities of the events as output of the event-based camera. (Left) Screenshot of one single event (in white) at t_i. (Middle) Timings since the latest event, or *time context*, at time t_i, forming the matrix $T(t_i)$ (white represents $-\infty$). (Right) Time surface at t_i as the matrix $TS(t_i)$ (note that the maximum of 1 is reached for the current event).

2.3 Event-based formalism: HOTS model

The HOTS model comprises three aspects. First, a core mechanism is defined that transforms any incoming event from the stream of events into a novel event as it is selected in a layer of neurons (see figure 3). This layer consists of perceptron-like neurons which measure the similarity of the input with patterns stored in the synaptic weights of the neurons. Crucially, this novel event is selected based on previous history thanks to the definition of what we will call *time surfaces*, and that will be used as the input to the current layer of the network. Second, the neuron which is inferred as being the most similar emits an output event at the same time as the incoming event. This core mechanism is defined on arbitrary address spaces and forms one layer of the network. Using it as a building block, such layers can be stacked together, each layer's output address space defining a novel input address space for the next layer. Finally, this constructs a *hierarchy* of layers organized in a feedforward fashion. Third, the core mechanism can be used in the particular case of the event streams produced by an event-based camera by defining a set of addresses relative to the pixel grid. For this, it reproduces the core mechanism in each layer at every position of the pixel grid. This defines weights as *kernels*, similarly as Convolutional Neural Networks (CNNs). Let's now formalize these three aspects independently.

2.3.1 Time Surfaces

The output of an event-based camera is a discrete stream of events (see figure 1) which can be formalized as an

ordered set of addresses: $\{a_i\}_{i\in[0,N_{ev})}$ where $N_{ev} \in \mathbb{N}$ is the total number of events in the data stream. On a camera for instance, each address is typically in the form $a_i = (x_i, y_i, \mathbf{p}_i)$, where (x_i, y_i) defines its position on the pixel grid and $\mathbf{p}_i$ its polarity. This formalism is defined over the address space $\mathcal{D}$. On a camera, we can define $\mathcal{D} = [0, N_X) \times [0, N_Y) \times [0, N_p) \subset \mathbb{N}^3$ where (N_X, N_Y) is the size of the sensor in pixels and N_p is the number of polarities. Each event is usually associated with a time t_i. We may now introduce the definition for the subset of events' ranks that occurred at or before a given time $t \in \mathbb{R}^+$ at a given address $a \in \mathcal{D}$:

$$\xi_a(t) = \{j \in [0, N_{ev}) | a_j = a, \text{and } t_j \leq t\}$$

Note that this definition is given for any continuous time t but is usually computed at the time of events.

For the corresponding stream of events occurring at the address a, it is possible to construct what is called a time context $T_a(t)$. It records the time of the latest event that occurred at that specific address a before or at t, with $-\infty$ if no event was recorded (see figure 2, middle column):

$$\forall a \in \mathcal{D}, \quad T_a(t) = \begin{cases} -\infty & \text{if } \xi_a(t) = \emptyset \\ \max\{t_i | i \in \xi_a(t)\} & \text{else.} \end{cases} \tag{1}$$

Finally, the time context $T_a(t)$ is computed for each address at any given time and thus forms a vector that we write $T(t)$ over the address space.

From the time context computed at each address, we finally derive the following set of values:

$$\forall a \in \mathcal{D}, \quad S_a(t) = e^{-\frac{t - T_a(t)}{\tau}} \tag{2}$$

where τ is a given time constant. This defines an analog vector over the address space that we call the *time surface* and that we write $S(t)$. In particular, it follows from the definition that $0 \leq S_a(t) \leq 1$ and that $\forall i$, $S_{a_i}(t_i) = 1$. An illustration of a time surface is given in figure 2, right column.

2.3.2 Architecture of the network: hierarchy

Let us now formalize the building block of the HOTS algorithm as a core mechanism defined on a neural layer. Specifically, let's consider that the layer is composed of N_n neurons which form a novel address space $\mathcal{A}$ that we may index as $\mathbf{n} \in [0, N_n)$. Each neuron is defined by a weight vector $W_\mathbf{n} = [w_{a,\mathbf{n}}]_{a\in\mathcal{D}}$. This vector has the dimension of the dendritic space associated to the input of that layer. These may be composed into a weight matrix $W = [w_{a,\mathbf{n}}]_{a\in\mathcal{D},\mathbf{n}\in\mathcal{A}}$. These weights will be used to compute the similarity of weight patterns with each time surface [35]. The similarity measure $\beta_\mathbf{n}$ is defined as the scalar product over the dendritic space $\mathcal{D}$:

$$\beta_\mathbf{n}(t) = \langle W_\mathbf{n}, S(t) \rangle = \sum_{a\in\mathcal{D}} w_{a,\mathbf{n}} \cdot S_a(t) \tag{3}$$

Whenever a new event enters the layer at time t_i, then this layer will emit one unique event with the same timestamp and with an address corresponding to that of the neuron which weight vector is the most similar to the time-surface as input:

$$\mathbf{n}_i = \arg\max_{\mathbf{n}\in\mathcal{A}} \beta_\mathbf{n}(t_i)$$

As a summary, this process thus transforms the list of input addresses $\{a_i\}$ into a novel stream $\{\mathbf{n}_i\}$ with identical timestamps $\{t_i\}$.

As stated above, this building block can be stacked by using the output address space to define the input address space of a subsequent layer. We will index layers by $\mathbf{L}$ and, to describe the input of a layer $\mathbf{L}$, we define a dendritic address space $\mathcal{D}^\mathbf{L}$ (with $\mathcal{D}^{\mathbf{L}=0} = \mathcal{D}$). We also set an axonal address space $\mathcal{A}^\mathbf{L}$ for the output of the layer. If we define $\mathcal{D}^{\mathbf{L}+1}$ based on $\mathcal{A}^\mathbf{L}$, then we can stack the different layers: the stream of event will cascade from the first to the last layer. Each layer will be defined by a weight matrix $W^\mathbf{L}$ such that each time surface will be associated with a similarity measure, which will generate events in the axonal address space $\mathcal{A}^\mathbf{L}$. Since each incoming event generates one and only one output event in each successive layer, we may compute for each incoming event a time surface at each layer. For this computation, we will use a different time constant $\tau^\mathbf{L}$ that will vary for each layer of the network. We will designate the corresponding time surfaces at each layer $\mathbf{L}$ as $S^\mathbf{L}(t)$. This process defines the core mechanism of the HOTS model.

2.3.3 Architecture of the network: kernels

Let us now define the topology of the address spaces. We saw that each time surface $S^\mathbf{L}(t)$ stores an analog value function of the delay between t and the last event that was recorded in the dendritic address space $\mathcal{D}^\mathbf{L}$. This value is then compared to weight vectors, similarly to the linear operation which is processed in the dendritic tree of perceptron neurons. However, from our knowledge on the early visual cortical areas, we know that the receptive field of neurons does not cover the whole visual space, but that they develop over limited visual space and with stereotyped shapes. This is used in CNNs to define different *kernels* which capture the local context in the neighborhoods around each neuron. From the spatial invariance of the physical problem one assumes that kernels should be similar across different positions and define a convolution operator. One notable advantage of this representation lies in its invariance to translations. In analogy with what is done in CNNs, we may thus define the connectivity of a HOTS core computation from this set of kernels that are translated on the sensor grid.

The dendritic address space for each layer is defined as follows: $\mathcal{D}^\mathbf{L} = [0, N_X) \times [0, N_Y) \times [0, N_p^\mathbf{L}) \subset \mathbb{N}^3$. The number $N_p^\mathbf{L}$ defines the number of channels of the time surface as input to the layer $\mathbf{L}$, we name them dendritic channels. Each address may be decomposed into its position and its dendritic channel, that is, $a^\mathbf{L} = (x^\mathbf{L}, y^\mathbf{L}, \mathbf{p}^\mathbf{L})$. The axonal address space of the layer $\mathbf{L}$ is $\mathcal{A}^\mathbf{L} = [0, N_X) \times [0, N_Y) \times [0, N_n^\mathbf{L}) \subset \mathbb{N}^3$, where $N_n^\mathbf{L}$ is the number of axonal channels. The similarity measure may thus be written as:

$$\beta^\mathbf{L}_{(x^\mathbf{L}, y^\mathbf{L}, \mathbf{k}^\mathbf{L})\in\mathcal{A}^\mathbf{L}}(t) = (\tilde{K}^\mathbf{L}_\mathbf{k} * S^\mathbf{L}(t))(x^\mathbf{L}, y^\mathbf{L}) \tag{4}$$

where $*$ is the convolutional operator and $\sim$ is the symmetric operator, which allows computing the correlation in equation (3) using the convolution. Note that $\tilde{K}^\mathbf{L}_\mathbf{k} * S^\mathbf{L}(t)$ represents the activity map and can be efficiently computed by a convolution operation. In our formalism, time surfaces are

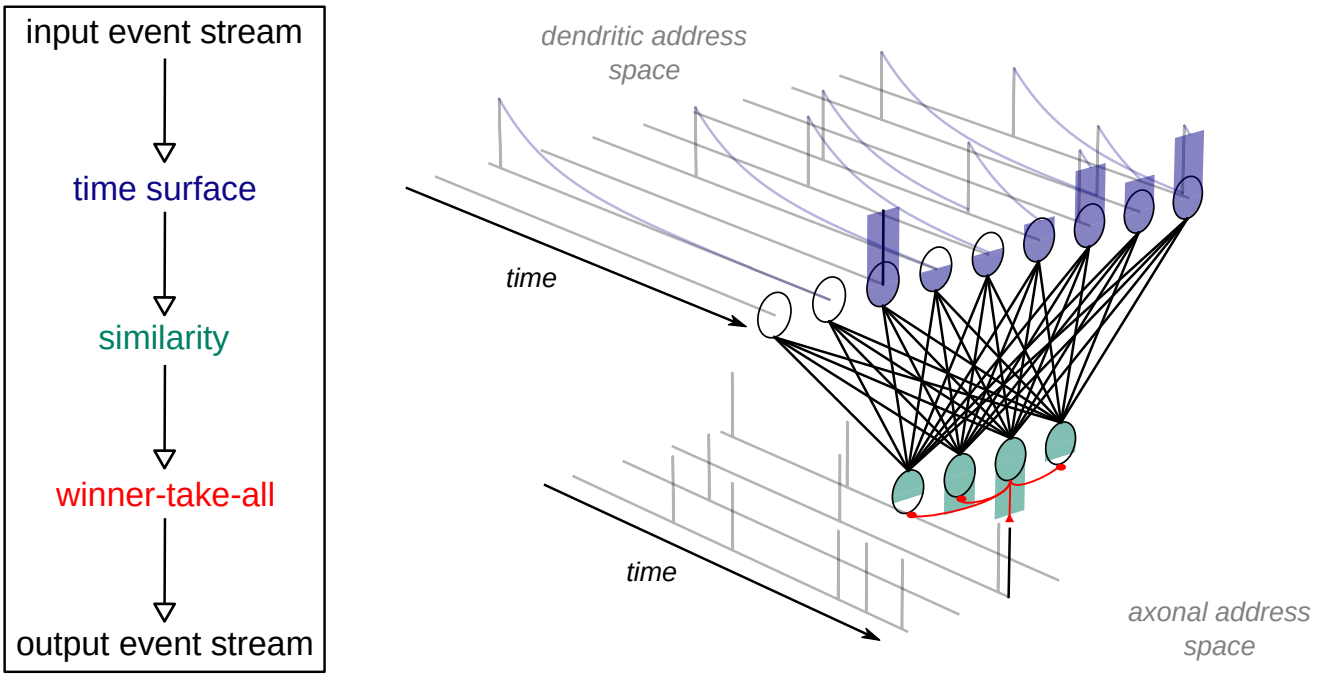

Fig. 3: Illustration of the core computation made within one layer of the HOTS algorithm. On the top of the plot, we show the dendritic stream of events convolved by an exponential decay which forms the time surface. Time surfaces are computed at the timestamp of each event/spike. The time surface at present is represented with the colored bar plot on the top. In the vertical slice, computations made within one layer at time t_i are illustrated. The time surface is compared to all the kernels of the layer with the similarity measure resulting in the membrane potential of the postsynaptic neuron represented in green. As an illustration, the layer contains only 4 neurons associated to 4 different kernels and with 10 dendritic inputs. At last, a winner-take-all rule (or $\arg\max$ non-linearity) will choose at time t_i the most activated neuron. This will emit a spike and prevent the others from being activated through lateral inhibitions (in red). Note that for each event as input of the layer, a new event will be emitted with the same timing as the incoming event.

defined globally and each weight vector corresponds to one column of the weight matrix, constructed with a Toeplitz operation, in which indices are associated to each axonal address: $(x^{\mathbf{L}}, y^{\mathbf{L}}, \mathbf{k}^{\mathbf{L}})$. The local context for the kernels is defined, on the topography of the pixel grid, by a radius $R^{\mathbf{L}}$ and on all channels of the time surface. The weights outside that radius are zero, and thus the similarity measured with the global time surface $S(t)$ will generate the same results as with the time surfaces defined locally in the original HOTS formalization.

Moreover, the HOTS algorithm, specified in [1], enforces that the position of each event is not changed from one layer to the next. As a consequence, each kernel still acts as a convolution kernel, but the comparison is only to be performed for the addresses corresponding to the position (x_i, y_i) of the event. This restriction can be implemented by defining the subset of output neurons with the exact same position but across the different axonal channels and modifies the match equation to:

$$\mathbf{p}_i^{\mathbf{L}+1} = \underset{\mathbf{k}^{\mathbf{L}} \in [0, N_n^{\mathbf{L}})}{\arg\max}\ \beta^{\mathbf{L}}_{(x_i, y_i, \mathbf{k}^{\mathbf{L}})}(t_i)$$

As a consequence, the next layer will emit an event $a_i^{\mathbf{L}+1} = (x_i, y_i, \mathbf{p}_i^{\mathbf{L}+1})$ with the same timestamp t_i, with the same spatial position (x_i, y_i) but with a different channel. As a summary, each layer takes input events from its previous layer and feeds events to the next one by reproducing these steps. It follows that neurons within a layer $\mathbf{L}$ are competing across features: each incoming event produces a single event on the axonal space. Following what is observed in biological visual pathways of mammals, we may set the number of axonal channels $N_n^{\mathbf{L}}$, the time constant $\tau^{\mathbf{L}}$ and the radius of the kernels $R^{\mathbf{L}}$ in such a way that these will increase when passing from one layer to the next. The choice made in the original HOTS algorithm is to double the radius of a kernel and the number of channels from one layer to the next, while multiplying the time constant from one layer to the next by a factor of ten. As a consequence, the network will learn more and more complex spatio-temporal features in a hierarchical way. We keep the same multiplication factor from one layer to the next one for the number of kernels $N_n^{\mathbf{L}}$ and for the radius $R^{\mathbf{L}}$. For the time constant $\tau^{\mathbf{L}}$, we set it as a function of the number of channels in the time surface, such that it is adapted to the average Inter-Spike Interval on each layer: $\tau^{\mathbf{L}} = N_p^{\mathbf{L}}.\tau^{\mathbf{L}=0}$. A description of the hyper-parameters associated to the experiments on the different datasets is reported in 3.1.

Regarding the learning of the weights, it is performed in an unsupervised fashion. During the unsupervised clustering phase, kernels are updated with the same learning rule as described in [1]:

$$\tilde{K}^{\mathbf{L}}_{\mathbf{p}_i^{\mathbf{L}+1}} \leftarrow \tilde{K}^{\mathbf{L}}_{\mathbf{p}_i^{\mathbf{L}+1}} + \eta_{\mathbf{p}_i^{\mathbf{L}+1}} \cdot \beta_{\mathbf{p}_i^{\mathbf{L}+1}} \cdot (S^{\mathbf{L}}_{local}(t_i) - \tilde{K}^{\mathbf{L}}_{\mathbf{p}_i^{\mathbf{L}+1}})$$

$$\text{with } \eta_{\mathbf{p}_i^{\mathbf{L}+1}} = \frac{0.01}{1 + \frac{\#_{\mathbf{p}_i^{\mathbf{L}+1}}}{20000}}$$

where we define $\#_{\mathbf{p}_i^{\mathbf{L}+1}}$ as the number of times kernel $\tilde{K}^{\mathbf{L}}_{\mathbf{p}_i^{\mathbf{L}+1}}$ was already selected and $S^{\mathbf{L}}_{local}(t_i)$ is the time surface defined locally, i.e. around the event as input of the layer with a radius $R^{\mathbf{L}}$. That is, once a neuron is matched, a Hebbian-like mechanism is used to take the selected kernel $\tilde{K}^{\mathbf{L}}_{\mathbf{p}_i^{\mathbf{L}+1}}$ closer to the observed time surface. Indeed, $\beta_{\mathbf{p}_i^{\mathbf{L}+1}}$ represents the product of activation for presynaptic and postsynaptic neurons. Note that the training of the kernels is shared among all the spatial locations for the same axonal channel, just like in CNNs. This mechanism is similar in principle to that used by the k-means algorithm and is implemented in numerous other unsupervised learning schemes [36]. For the layers of the HOTS model, we filter time surfaces with a threshold on the number of active pixels such that we avoid noisy or isolated events. We set this threshold to $2 \cdot R^{\mathbf{L}}$. Figure 4 offers an illustration of the different kernels learned by the network.

2.4 Homeostasis

The contribution of homeostasis to the robustness of the HOTS model is the guideline of a previous work [2]. Similar regulation methods on an event-based dataset are used in [37, 38] to balance the firing rate over the neurons of each layer of the SNN. The model of [37] uses an adaptive membrane threshold, while [38] adds an auxiliary neuron per layer for regulation of neurons' firing rate. In this last paper, they make a comparison of this technique with zero-mean batch normalization [39] used for training deep neural networks. These methods are similar in their objectives and are well justified in terms of efficient coding [40].

Here, we implement homeostasis regulation by adapting the heuristics used in a sparse coding scheme [41]. It simply consists in modifying the similarity measure (see equation (3)) as:

$$\beta_{\mathbf{k}}(t) = \gamma_{\mathbf{k}}(t) \cdot \langle W_{\mathbf{k}}, S(t) \rangle \tag{5}$$

Where we use the same gain as defined in [2]:

$$\gamma_{\mathbf{k}}(t) = e^{\lambda \cdot (f_{\mathbf{k}}(t) - \frac{1}{N})} \tag{6}$$

where λ is a regularization parameter, $f_{\mathbf{k}}$ is the relative activation frequency of kernel $\mathbf{k}$ and N the total number of kernels in that layer. Note that the gain control is applied on each map of kernel activities, not the individual neurons' activity, due to the translation invariance property of the architecture. This regulation rule allows training the different kernels such that it avoids the response of only a few of them, and it reaches equilibrium when $f_{\mathbf{k}}(t) = \frac{1}{N}$, that is, when they are on average activated equi-probably.

In practice, we observed that adding homeostasis leads to a better clustering of the weight matrices, see figure 4. Note that neural activity is balanced during the unsupervised clustering phase across all digits. The homeostasis process does not necessarily yield an equi-probable neural activity for one digit, but over the whole learning set, in line with the efficient coding hypothesis [42]. In addition, it allows avoiding introducing an *ad hoc* heuristics in the learning rule to reach convergence for all neurons. In [1] for instance, weight matrices or synaptic weights associated with each neuron were initialized with the first incoming time surfaces. The original method makes the learning of weight matrices very sensitive to initialization. In addition, the hierarchy is learned sequentially, one layer after the other. In this work, weights were instead initialized at random, and we allow spikes to feed each layer of the network even if a given layer is not fully trained, yet convergence was reached robustly. As a result, this additional ingredient in the unsupervised learning phase makes our algorithm behave more similarly to the conditions faced by living systems.

2.5 Online event-based classification

In the original HOTS algorithm [1], classification is performed by comparing the activation histograms across the channels of the last layer of the network to that observed on average for each given class. This classification with histogram comparison is performed *post hoc*, after the coding of an item from the dataset. Here, we introduce a novel online classification scheme, that is, where classification is performed for each spike reaching the classifier, and more generally at any time when a classification is necessary. Following the same strategy used for the construction of time surfaces, each event reaching the last layer $\mathbf{L} = \mathbf{C}$ of the network may indeed be transformed into a time surface $S^{\mathbf{C}}(t)$ using a time constant $\tau_{\mathbf{C}}$. This constant may change from one dataset to another according to the statistics of the samples. The time surface thus forms an analog vector that may be used in a Multinomial Logistic Regression (MLR) model to achieve supervised classification. Such MLR models are for instance used in the last layer of classical deep-learning networks [43] and are compatible with a neural implementation [44]. More specifically, it corresponds to the similarity measure (see equation (3)) of the MLR weights with the input, stacked with a sigmoid non-linearity. The weights are defined on the whole dendritic space, that is, there is no local context as it was defined for the kernels on the previous layers. For each event, the output neurons will compute the probability of predicting the respective class. In the MLR, this probability value is computed as a softmax function of the linear combination of the analog vector as input:

$$\forall c \in \{1, \ldots, N_{\text{class}}\},$$
$$Pr(y = c | t_i; W^{\mathbf{C}}) = \frac{e^{\langle W^{\mathbf{C}}_c, S^{\mathbf{C}}(t_i) \rangle}}{\sum_{j=1}^{N_{\text{class}}} e^{\langle W^{\mathbf{C}}_j, S^{\mathbf{C}}(t_i) \rangle}}$$

where $W^{\mathbf{C}}_j$ are the coefficients associated to class j of the MLR model. As in section 2.3.2, the formulation of the time surface can be extended to the continuous time domain. It follows that the probability value can be computed at any time when necessary. We simplify the notation of the probability value by defining the following equation for the softmax function:

$$\sigma_c(t) = \frac{e^{\beta^{\mathbf{C}}_c(t)}}{\sum_{j=1}^{N_{\text{class}}} e^{\beta^{\mathbf{C}}_j(t)}} \tag{7}$$

Where $\beta^{\mathbf{C}}_c(t)$ is the similarity measure (from equation (3)) between the time surface as input to the classification layer and the weights of the MLR model associated to the class c. The final prediction can be made for each incoming event

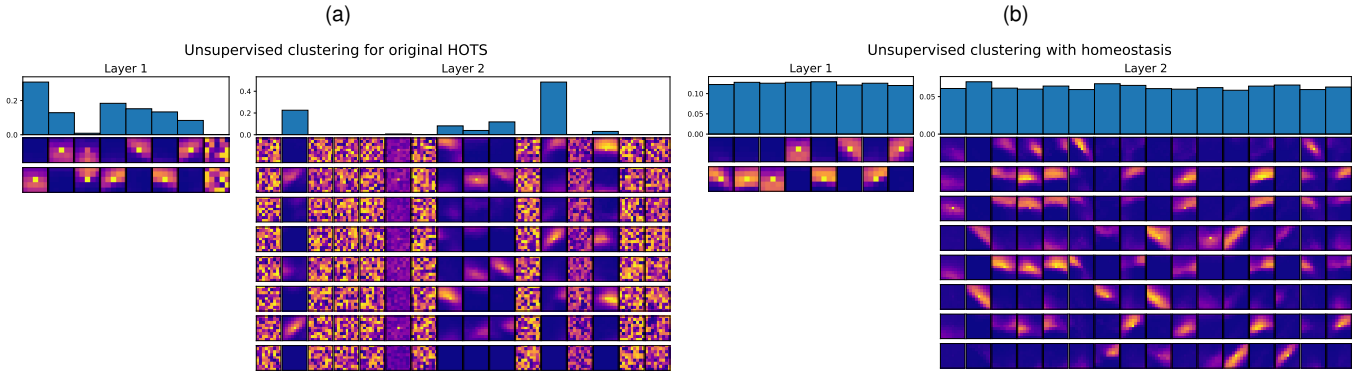

Fig. 4: Activation histograms and time surfaces obtained in the unsupervised learning algorithm (a) for the original HOTS network (replicated from [1] with time surfaces intialized randomly) and (b) for the bio-plausible version with homeostasis. Activation histograms correspond to the frequency by which each neuron was activated. For each layer number n, $f_n = \frac{1}{N_n}$ is the averaged activation frequency. Associated time surfaces are plotted below histogram bins. The different lines are the different polarities of the features (ON and OFF for the first layer), that is, the output neurons of the previous layer for the next one.

with the $\arg\max_c$ function by selecting the class associated with the highest probability. Then, thanks to the definition of the softmax function, we obtain its maximum value by getting the maximum value of the similarity measure. We obtain the same spiking process as in any layer of the HOTS network:

$$c(t) = \underset{c\in\{1,\dots,N_{\text{class}}\}}{\arg\max}\ \sigma_c(t)$$

It results in an always-on decision process, able to make a prediction at any time. In the following, we will perform event-driven prediction and compare the results of the classification as a function of the number of events fed to the classifier or as a function of time. Using this probabilistic formalism, we can also provide predictions with higher confidence to improve the performance of the classification. Even if the probability values are computed for each event, the prediction of the class can be done only for some events with a probability over a defined threshold. This flexibility in making predictions with a specific confidence threshold allows improving performances while keeping an event-driven computing approach.

In practice, we first trained the hierarchical network using unsupervised online learning on a training set. On this set, we computed the transformation of the input stream into the output stream and then transformed it into time surfaces to feed the classification layer. We trained the MLR model using, as supervision pairs each time surface along with its true class. The MLR model was implemented with the PyTorch language, and training was performed by a gradient descent with the Adam optimizer. Our loss function is the binary cross entropy computed on the output spike train, and learning parameters are described in 3.1. Once the MLR model was trained, we obtained analog vectors from the computations of the hierarchical network on the testing set. Then, we tested classification performances by sending these vectors to the MLR model, which outputs the probability for each class to be true. The decision process can be a $\arg\max$ function of the probability values, and it allowed us to compute an accuracy on an event-by-event basis.

2.6 The Spiking Neural Network analogy

We have defined the HOTS algorithm in an event-based formalism, and we demonstrate that when it is extended to the continuous-time domain, this algorithm may be implemented as a SNN. Indeed, the definition of the time surface modulated by an exponential decay in equation (2) bears an analogy with the LIF model with exponentially decaying postsynaptic potentials, as described for other SNNs [45]. We aim at describing the event-based model on a time continuum thanks to Ordinary Differential Equations (ODE) and bridge such an algorithm with the SNN framework from computational neuroscience.

2.6.1 HOTS as a SNN

Let's consider the fundamental mechanism of the HOTS algorithm at some layer $\mathbf{L}$ (we will omit this superscript for clarity in this section). In the previous section, time surfaces are defined at any time using equation (2). By looking at figure 3, one can observe that the dendritic addresses refer to the presynaptic neurons and that the temporal kernel defined by the time surface corresponds to the Spike Response model [46] of a first-order linear ODE. Each presynaptic neuron corresponding to an address $a \in \mathcal{D}$ received the events with ranks from the set $\xi_a(t)$ and the evolution of $S_a(t)$ thus follows the ODE:

$$\frac{\mathrm{d}}{\mathrm{d}t}S_a(t) = -\frac{1}{\tau}\cdot S_a(t) + \sum_{i\in\xi_a(t)} (1 - S_a(t))\cdot\delta(t-t_i) \quad (8)$$

The second term of the right side of equation (8) is a modulated Dirac function which implements the integration of a novel presynaptic potential at $t = t_i$. The modulation $1 - S_a(t)$ is such that at the moment of the event as the new value of the potential becomes $S_a(t) + (1 - S_a(t)) = 1$. It thus implements the fact that the maximum value of a time surface is equal to 1, and that only the time until

the last spike has an influence on activity, as implemented in the definition of the time context. As a consequence, it implements a form of resetting mechanism which allows computing the time surface as a function of the time to the last spike.

Then, for a postsynaptic neuron $\mathbf{n}$ of the layer, we may define a membrane potential which corresponds to the integration of synaptic inputs into the similarity measure:

$$\beta_{\mathbf{n}}(t) = \langle W_{\mathbf{n}}, S(t) \rangle = \sum_{a \in \mathcal{D}} w_{\mathbf{n},a} \cdot S_a(t)$$

where we use the same weights $W_{\mathbf{n}}$ of equation (3) from the event-based formalism. Finally, by integrating over the different input synapses, we get a differential equation that describes the dynamics of the membrane potential $\beta_{\mathbf{n}}$ as a similarity measure:

$$\frac{\mathrm{d}}{\mathrm{d}t}\beta_{\mathbf{n}}(t) = -\frac{1}{\tau} \cdot \beta_{\mathbf{n}}(t) + \sum_{a \in \mathcal{D}} w_{\mathbf{n},a} \cdot \sum_{i \in \xi_a(t)} (1 - S_a(t)) \cdot \delta(t - t_i)$$

Which may be simplified into a sum over all events:

$$\frac{\mathrm{d}}{\mathrm{d}t}\beta_{\mathbf{n}}(t) = -\frac{1}{\tau} \cdot \beta_{\mathbf{n}}(t) + \sum_{i=0}^{N_{ev}} w_{\mathbf{n},a_i} \cdot (1 - S_{a_i}(t)) \cdot \delta(t - t_i)$$

Such a ODE is classical for the description of the evolution of the membrane potential of LIF neurons. Note that the major change lies in the modulation of the integration of incoming spikes, which allows representing only the time until the last spike. The hierarchical network proposed in [1] is then equivalent to a SNN composed by LIF neurons with a Hebbian-like learning mechanism, as mentioned in section 2.3.3. In this SNN, for every incoming event from the event-based camera, one spike is emitted for each layer of the network. It results in a winner-take-all (WTA) competition between neurons within the same layer.

2.6.2 MLR as a SNN

The classification layer of our algorithm is defined as a MLR model, for which a parallel with a SNN implementation was already drawn in [44]. Analogous to biology and as described in section 2.6.1, the linear combination of the time surface as input with the MLR weights corresponds to the integration of presynaptic spikes on the dendritic tree of one postsynaptic neuron associated to one class. Then, $\beta_c(t) = \langle W_c^{\mathbf{C}}, S(t) \rangle$ represents the membrane potential of the postsynaptic neuron associated to class c and $W_c^{\mathbf{C}}$ are the corresponding synaptic weights. The softmax function presented in Equation (7) is a good model of a spiking WTA network. Indeed, Nessler et al. [47] demonstrated that a stochastic spiking WTA can be built from this type of activation function. The denominator expresses the lateral inhibition by the other neurons of the layer. The $\arg\max_c$ function imposes a full inhibition of other neurons until the next decision. As a consequence, if the classification is event-driven, only one spike is emitted for the most probable class only at each event. Then, the spiking mechanism of the classification layer is the same as for the rest of the network due to the fact that the logistic function is monotonic.

The main difference with the other layers of the network lies in the supervised learning rule of the MLR weights. We can obtain the learning rule by finding the derivative of the loss function. For the softmax regression, the loss function for a rank i event is the binary cross-entropy:

$$J(t_i) = -\sum_{c=1}^{N_{\text{class}}} \delta_{\{y(t_i)=c\}} \cdot \log(\sigma_c(t_i))$$

where $\delta_{\{y(t_i)=c\}}$ is the 'indicator function' and $y(t_i)$ is the true class. If we compute the derivative of the loss function with respect to $W_c^{\mathbf{C}}$, we can obtain the update rule of the weights of the postsynaptic neuron associated to class c:

$$\Delta W_c^{\mathbf{C}}(t_i) = \begin{cases} \eta \cdot S^{\mathbf{C}}(t_i) \cdot (1 - \sigma_c(t_i)), & \text{for } c = y(t_i) \\ -\eta \cdot S^{\mathbf{C}}(t_i) \cdot \sigma_c(t_i) & \text{for } c \neq y(t_i) \end{cases}$$

where η is the learning rate. This correlation-based learning rule can be described as a supervised Hebbian learning mechanism, with different possible weight updates depending on the true value of the outcome.

In summary, the event-based algorithm that we use in this work can be fully described by a SNN. Learning of the weights is performed in an event-driven fashion and corresponds to Hebbian-like mechanisms for the neurons, inside the network and in the classification layer as well. We claim that these local learning rules are an advantage both in terms of bio-plausibility and for energy-efficient on-chip implementations [48].

3 Results

In this section, we present the classification results we obtain with our method. We start by presenting the online classification performance of the network, which is the main novelty of our study. Then, we compare performances to the state-of-the-art (SOTA) on the different datasets by reporting the accuracy obtained when making one prediction per sample. We complete our analysis by studying the resilience of our algorithm to both temporal and spatial jitter and compare it to the original method proposed in [1].

Let's first give a detailed description of the parameters and the architecture of the networks tuned for classification on the different datasets. We report these parameters in table 1. Parameter tuning for $\mathbf{L}_1$ and $\mathbf{L}_2$, which are layers similar to the ones in the original HOTS network, was done on subsets from the different datasets by computing the accuracy for each of the different architectures with histogram comparison as done in [1]. For every dataset, the classification layer is implemented in PyTorch, and we train it with gradient descent and the Adam optimizer. Time surfaces as input of this last layer are defined globally, i.e. on the whole pixel grid, and we adjust the time constant for each dataset. Time constants are also obtained empirically, by testing the performances of the classifier with different parameters, on a subset of the original dataset. We set, for PokerDVS dataset, the number of epochs to 33 and keep the default parameters of the optimizer. For both N-MNIST and DVSGesture datasets, to reduce the number of computations and avoid reaching a local minimum within the first samples, we perfom the learning only on a randomly-chosen percentage of the computed time surfaces. For N-MNIST, we keep 10% of the time surfaces of a sample and have the same number of epochs: 33. For DVSGesture, we keep only 5% of time surfaces per sample and increase the number of epochs

to 65. Table 1 reports the different time constants, learning rate used for training and the threshold on the probability values used to compute performances of the classification.

	$\mathbf{L}_1$	$\mathbf{L}_2$	**MLR**
Poker DVS	$N_K = 8$ $R = 2$ $\tau = 1$ ms	$N_K = 16$ $R = 4$ $\tau = 4$ ms	$\eta = 0.005$ $\tau_{\mathbf{C}} = 30$ ms $\theta = 0.9$
N-MNIST	$N_K = 16$ $R = 2$ $\tau = 20$ ms	$N_K = 32$ $R = 4$ $\tau = 160$ ms	$\eta = 0.005$ $\tau_{\mathbf{C}} = 50$ ms $\theta = 0.99$
DVS Gesture	$N_K = 16$ $R = 4$ $\tau = 100$ ms	$N_K = 32$ $R = 8$ $\tau = 800$ ms	$\eta = 0.0001$ $\tau_{\mathbf{C}} = 100$ s $\theta = 0.9$

TABLE 1: Parameters of the network. $\mathbf{L}_1$ and $\mathbf{L}_2$ are the core unsupervised layers where we report the number of kernels (N_K), the size of the receptive fields (R) and the time constants (τ) associated to each layer. **MLR** is the supervised classification layer trained with a specific learning rate η, time constant $\tau_{\mathbf{C}}$ and a threshold on the output to make the decision θ.

3.1 Online inference

We first present the results of the end-to-end event-driven online classification described in section 2.5. To illustrate the dynamic evolution of the event-based classification performance, we plot the value of the accuracy as a function of the number of events received by the classifier for each dataset (see figure 5). We present two decision-making modes for the classifier. The first is *online HOTS*, where a prediction is made for each incoming event without any conditions on the probability values corresponding to the different classes. The second is *online HOTS with threshold*, i.e. when the output of the classifier must reach a probability threshold, reported for each dataset in table 1, to make a decision. In this last condition, in order to filter out events in periods with low information (especially at the beginning), we select events with a confidence of the decision above a threshold. This improves the average performance of the classification. However, it introduces a small time lag in order to accumulate enough evidence to make a prediction. We also report the accuracy values for the classification *post hoc* with comparison of activation histograms, as in [1]. Two results are mentioned in figure 5, one for the original *HOTS* as in [1] and another with the homeostatic gain control for the clustering phase: *HOTS with homeostasis*.

As expected, for all datasets and both modalities, the accuracy of the online classification improves as the number of events increases. Within a dataset, the total number of events for the samples can be different. We set a maximum number of events to represent the accuracy by taking the 90 percentile of the dataset in terms of number of events. The accuracy of the event-based classification for each dataset is presented in figure 5. Note that the x-axis is plotted with a logarithmic scale and that only a small number of events allows for significant classification above chance.

In figure 5-(a), we observe the online inferences for the Poker DVS dataset. The RESULTS_PokerDVS.ipynb notebook reproduces the results and figures for this dataset. The *post hoc* methods perform well but do not reach 100% accuracy, with an advantage for clustering with homeostasis (95.0% accuracy) than without (85.0% accuracy). For *online HOTS*, the accuracy quickly reaches 100% after only an average of 19.4% of the total number of events, i.e. one fifth of the complete event stream. This online classification allows an ultra-fast categorization of objects in terms of events: only a few events are needed for the classification to reach a good level of accuracy. Given the small number of samples in this dataset, we evaluate the performance of the network on two more complex and widely used datasets.

Online accuracy on the N-MNIST dataset is given in figure 5-(b) and reproducible at RESULTS_NMNIST.ipynb. The original algorithm already performs well with the classification by histogram comparison, reaching 94.4% for *HOTS* and 92.4% for *HOTS with homeostasis*. In this particular example, the homeostatic gain control did not improve performances for the clustering phase, and we remind that the main advantage of this regularization is to reduce the sensitivity of the unsupervised learning to initialization [2]. For *online HOTS*, we observe an accuracy above chance after the very first events that keeps increasing with the number of events received by the network. Note that the accuracy value increases drastically after approximately 1000 events and reaches values above the original method at 2000 events on average, the mean number of events for the N-MNIST dataset being 4176 events (see DATASET_STATS.ipynb). If we compute the mean performance over all the decisions, i.e. for each event, we obtain an accuracy of 70.1% and 96.6% for the accuracy computed when the decision is taken at the timestamp of the last event. Another way to compute the *post hoc* accuracy with this probabilistic approach is to choose the decision that was made with the highest confidence. By doing so, we obtain an accuracy of 97.4%, close to the SOTA (see the following section). We also show the flexibility and the advantage of using this MLR model by setting a minimum likelihood value, necessary to make a decision (see the *online HOTS with threshold* curve in figure 5-(b)). With a threshold set at 0.99, good results can be obtained only after an average of about 100 events, in line with the idea of ultra-fast categorization. With this last method for decision-making, we obtain an average accuracy of 96.2%. This algorithm shows engaging performances for ultra-fast digit recognition. The results of 5-(b) indicate that the second and the third saccade of the N-MNIST recordings add only a small amount of information, and the evolution of the accuracy in figure 5-(b) illustrates this point. Previous works report accuracy results using only the first saccade and show only small improvement when using the other ones as well [24, 49, 27].

For the DVSGesture dataset, we confirm the improvement of our method compared to the original one on more realistic event-based recordings (see figure 5-(c)). For these more complex gesture recognition tasks, the *online HOTS* accuracy remains close to the chance level for at least 100 events. More evidence needs to be accumulated, and then the accuracy increases monotonously and outperforms the previous method after approximately 10.000 events (an average of 9.3% of the sample). These event-based recordings contain a much higher event density than for the other datasets, and we remind the reader that only 1 second of the recording is kept to test our algorithm. The mean

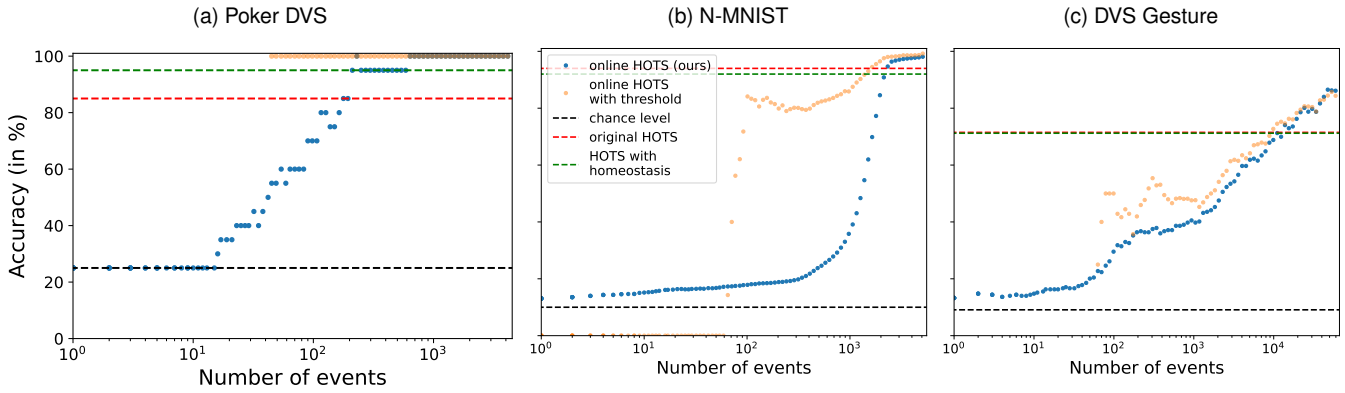

Fig. 5: Accuracy for online classification on 3 different datasets (see text for details).

accuracy for all the decisions is 73.3% and when we make a decision *post hoc*, choosing the classifier output with the highest probability, we obtain 78.8%.

3.2 Comparison to the state-of-the-art

	N-MNIST	DVS Gesture
HOTS (with k-NN) [1]	94.39%	71.18%
HATS [34]	99.1%	–
SLAYER [23]	99.2%	93.64%
DSNN-STDP [27]	95.77%	–
self-BP [30]	–	84.76%
hybrid CNN-SRNN [33]	–	97.61%
SCNN-STDP [32] (preprint)	99.26%	92.5%
Ours	**97.4%**	**78.8%**

TABLE 2: Offline classification accuracy for N-MNIST and DVSGesture datasets. The top part of the table corresponds to non-biologically plausible algorithms and the bottom part is for bio-plausible ones.

To compare the performances of our method with the SOTA, we choose to compute the accuracy when the decision is made with the highest confidence because other methods do not present the event-driven online accuracy in their results. We report a table of the best accuracy results found in the literature for the N-MNIST and the DVSGesture datasets. All methods mentioned in 2.2 are not reported here, preprint are discarded and we focus on event-based methods obtaining best performances. We split the table 2 into two distinct parts for methods that are biologically plausible (bottom) and the others (top). We skip the comparison of the results obtained with the Poker DVS dataset that acts as a toy model but does not provide a challenging classification task. We argue that, even if we don't outperform these SOTA results, this simpler 3-layered feedforward network structure with a bio-plausible learning obtains very competitive accuracy values. In addition, our classifier is the first to provide always-on decision-making.

3.3 Robustness to jitter

We also wanted to assess the robustness on this event-driven object recognition method. For this, we perturb the original datasets by adding temporal or spatial jitter to the events. Jitter is applied only on the testing set to add noise to the signal used for classification. As described in section 2.1, we use the *tonic* package to apply temporal or spatial jitter on the tested samples. For each amount of jitter applied to the testing set, 10 repetitions are made to get different accuracy values. Finally, we fit a beta distribution to each of these results to compute the percentiles plotted in figure 6. To compare with past results obtained in [2], we plot the offline accuracy obtained when making one decision per sample. To reduce the number of computations on this analysis, we use subsets of the N-MNIST dataset (1000 samples). For each amount of jitter applied to the testing subset, 10 repetitions are made to get different accuracy values. Finally, we fit a beta distribution to these results to compute the percentiles plotted in figure 6. Because the method we propose is a proof of concept for event-based computations, simulations on GPU are not optimized and results with jitter applied on the DVSGesture dataset are not computed. We highlight the fact, that to our knowledge, no other studies provide this test on event-based recordings so far. Simulated on two different DVS datasets, these results provide insights on the robustness of our algorithms but also highlights the features which are relevant for classification within the event-based recordings.

As expected, the higher the jitter, the stronger its negative impact on classification. The drop in accuracy as a function of jitter fits well a sigmoid function decreasing from a maximal accuracy value to reach chance level. Using this fit, one can define a critical standard deviation of jitter in pixels or in ms where accuracy drops to half its maximal value compared to chance level. This half-saturation level reveals a signature value for the relevant information contained in the signal. When no jitter is applied, *HOTS* and *HOTS with homeostasis* get comparable results. *online HOTS* performs significantly better than both methods for all amounts of jitter. We will now detail these results for each dataset.

Figures 6-(a, c) correspond to the PokerDVS dataset, while figures 6-(b, d) are results obtained on the N-MNIST dataset. One can notice a difference in variability of the results across the different trials. We explain this difference by the reduced number of samples in the PokerDVS dataset.

In figure 6-(a), we show the mean accuracy for the different methods as a function of the amount of spatial jitter

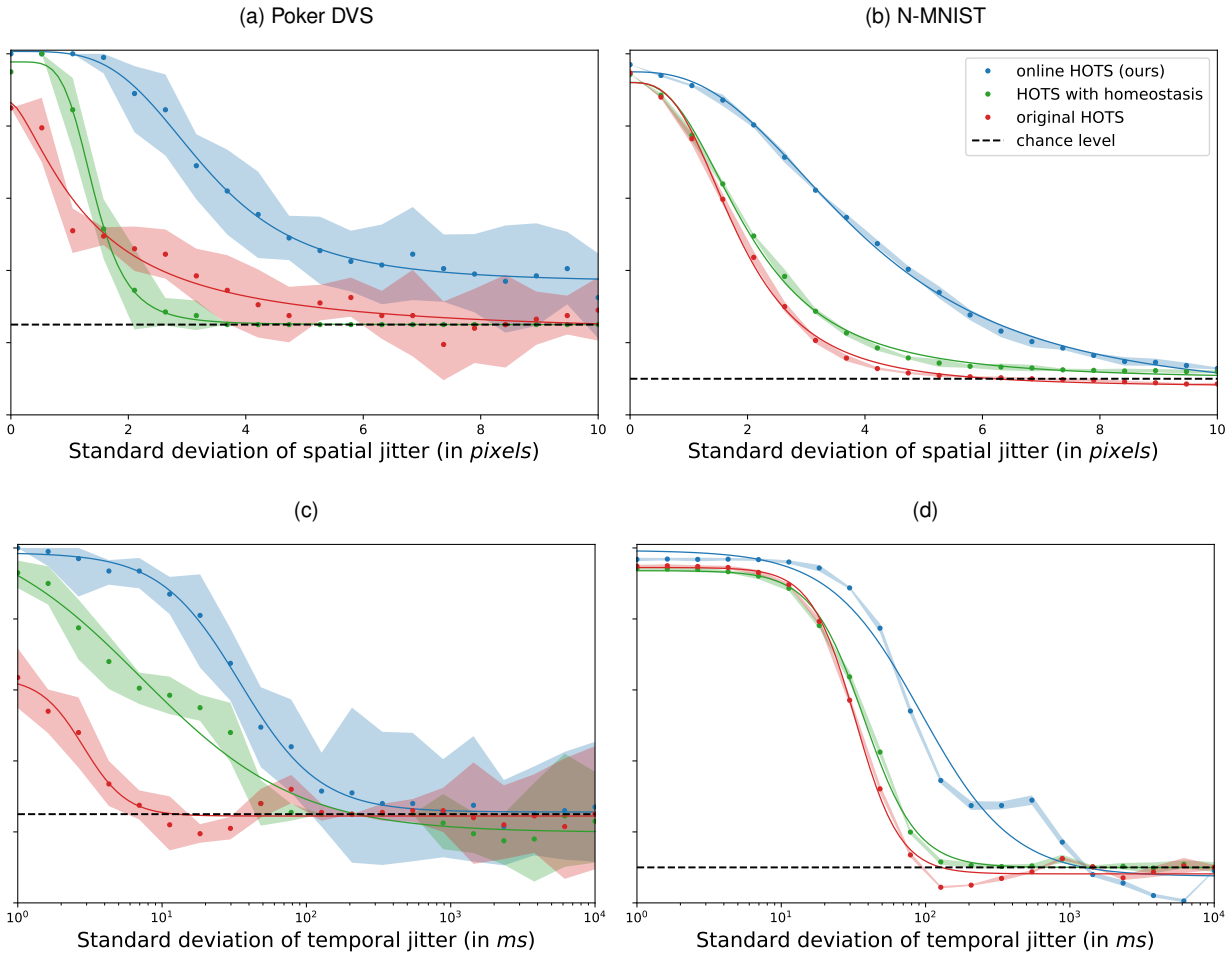

Fig. 6: Evolution of classification accuracy as a function of (a-b) spatial and (c-d) temporal jitter.

applied to the PokerDVS dataset. Accuracy reaches its half-saturation level for a spatial jitter with a standard deviation equal to 1.74, 1.56 and 3.51 pixels, respectively for *HOTS*, *HOTS with homeostasis* and *online HOTS*. In figure 6-(b), similar results are obtained for the N-MNIST dataset. There is less variability in the classification performances across the different trials and half-saturation levels are obtained with a standard deviation of 2.02 pixels for *HOTS*, 2.22 for *HOTS with homeostasis* and 4.14 for *online HOTS*. With the online classifier and for both datasets, robustness to jitter is increased significantly and the half-saturation level is doubled compared to the one from the original method.

Panels (c, d) in figure 6 illustrate a high resilience of the network to temporal jitter, note that the x-axis is composed of log_{10}-spaced values. Average recording time for the PokerDVS dataset is 7.1 ms and 308 ms for N-MNIST. For PokerDVS (see figure 6-(d)), we obtain the following half-saturation levels corresponding to a standard deviation of the jitter distribution in ms. For *HOTS*: 2.97 ms; *HOTS with homeostasis*: 10.1 ms; and *online HOTS*: 33.2 ms. For N-MNIST (see figure 6-(d)), the half-saturation values are 37.65 ms, 45.35 ms and 114.9 ms respectively for *HOTS*, for *HOTS with homeostasis* and for the algorithm presented in this study. Even the original method offers a high resilience to temporal jitter in comparison with the duration of the recordings. Then, this resilience increases with the addition of the homeostatic gain control and, for the *online HOTS* method, the standard deviation of jitter added needs to reach similar timescale as the recording itself (3 times the average duration of a recording for PokerDVS and a third of the average duration for N-MNIST samples). This robustness can originate from the use of time surfaces for signal encoding. When adding temporal jitter, locations of events are kept intact and only the timing is impacted. From the jittered signal, a time surface is computed with the same spatial structure by applying an exponential decay on the delays. This transform diminishes the impact of jitter. Then, the time surface is compared to smooth time surfaces with a scalar product on the whole spatial window. This technique renders the encoding of input events more robust to local temporal variations. The increase of resilience for the methods with a homeostatic gain regulation can also come from the improved clustering of the time surfaces of the network. The way we build the analog vector as input of the MLR layer can explain this surprisingly high resilience. Given the relatively high time constant used for the exponential decay (see Table 1), the combination of only few events on precise spatial locations can lead to a good prediction of the class. With this exponential decay, higher temporal resolution is achieved for events closer in time to

the one at which the time surface is computed. The higher it is, the higher the resolution for the recent past history, but the more events can accumulate on the same 2D time surface, disturbing the precise classification.

4 Discussion

In this study, we extended a neuromorphic engineering method with techniques inspired by computational neuroscience to develop an online, event-driven classification algorithm. We started our study with the HOTS network, whose original basis is inspired by the hierarchy found in the visual cortex. As designed in this network, the size of the receptive field increases along the visual hierarchy [50]. Furthermore, it was observed that cortical areas follow a hierarchical order of intrinsic time scales [51]. One hypothesis is that shorter time scales may be useful for rapid detection or tracking of dynamic stimuli, while longer time scales may be used for decision-making calculations performed by higher-level areas. This particular organization of the HOTS architecture and the evolution of the parameters of the temporal surfaces through the different layers follows physiological principles.

Furthermore, we show that our model is similar to that of an SNN by extending the equations to the continuous time domain. We present this unified theoretical framework to bridge the gap between neuromorphic engineering methods and computational neuroscience. We extend the event-based algorithm to a more generic and bio-plausible model. First, we used a homeostatic rule inspired by living systems to make the unsupervised online learning of the network more robust. Second, we added an online classification layer that performs MLR and is compatible with a neural implementation [44]. As demonstrated in section 2.6, the learning rules are local and correspond to Hebbian learning. This makes the learning of the network easily transferable to neuromorphic hardware. Once the network has been learned, it can perform an always-on classification, which means that, whenever necessary, a prediction can be inferred. We present the results obtained with event-based categorization, i.e. a prediction is made for each input event of the classification layer. There is no need to wait for the end of the recording of the sample or to collect a defined amount of events, which allows for ultra-fast categorization. This dynamic classification, which evolves over time for each new event, is closer to the object recognition performed by biological systems. We also demonstrate the advantage of using a probabilistic approach to classification by presenting the decisions made when a defined confidence threshold is reached. Although using a high confidence threshold to make a decision improves the overall classification performance, the classifier needs to accumulate more evidence to be able to categorize an event. The flexibility offered by this approach makes the algorithm a viable model for solving different tasks requiring fast or accurate decisions. Overall, these results provide a good illustration of the possible synergy between neuromorphic engineering and computational neuroscience.

For the datasets presented in this paper, only a reduced number of events in the sample were needed to reach good accuracy. We also observed a surprisingly high robustness of the method to temporal jitter. We infer that, for these datasets consisting of simple symbols, and because spatial locations are transferred along the different layers, the MLR model can capture specific relative locations of the output spikes to make sufficiently good decisions. To evaluate this assumption, we applied spatial subsampling to the global time surface that serves as input to the classifier. The MLR model is then learned and tested with the spatial resolution of the pixel grid, in this case 34×34, progressively reduced by the spatial subsampling. The results for the N-MNIST dataset are shown in Figure 7 and, together with the results in Figure 6-(b), highlight the importance of spatial information in the event stream, as shown by the classifier output. In comparison, the classifier does not seem to be sensitive to the temporal order of the events, but we believe that this may be due to the characteristics of the dataset and the use of time surfaces. Indeed, the choice of the time constant for the exponential decay is a trade-off between temporal accuracy for recently recorded events and the accumulation of events on the time surface. Therefore, we specifically aim in the future to develop a model capable of capturing accurate spatio-temporal patterns embedded in event-based recordings by modifying the input of the different layers of the network.

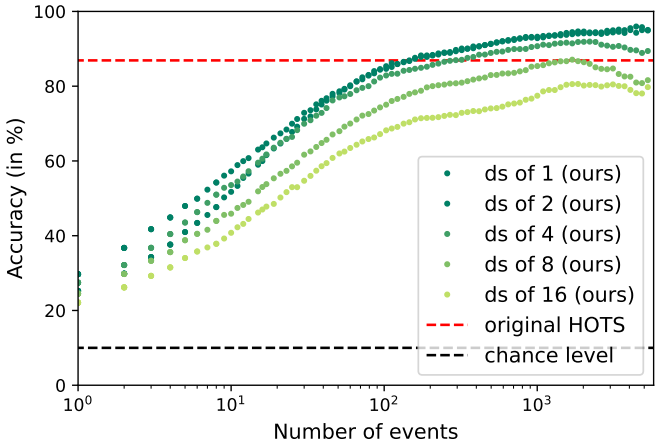

Fig. 7: Effect of the spatial downsampling (ds) on online classification accuracy as a function of the number of events on the N-MNIST dataset. MLR with a ds (downsampling) of 1 corresponds to the method presented in this paper and as the downsampling increases, the spatial resolution of the pixel grid decreases. We observe a significant effect of spatial resolution on classification performances, showing that the information necessary for object recognition is not fully captured by the HOTS network and is highly dependent on the spatial information and highlight the importance of spatial information in the stream of events as given by the output of the classifier.

Acknowledgment

Antoine Grimaldi and Laurent Perrinet received funding from the European Union ERA-NET CHIST-ERA 2018 research and innovation program under grant agreement N° ANR-19-CHR3-0008-03 ("APROVIS3D"). Sio-Hoi Ieng,

Ryad Benosman and Laurent Perrinet received funding from the ANR project N° ANR-20-CE23-0021 ("AgileNeuroBot").

REFERENCES

[1] X. Lagorce et al. "HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.7 (2017), pp. 1346–1359.

[2] A. Grimaldi et al. "A homeostatic gain control mechanism to improve event-driven object recognition". In: *Content-Based Multimedia Indexing (CBMI) 2021*. Apr. 16, 2021.

[3] T. Serrano-Gotarredona et al. "Poker-DVS and MNIST-DVS. Their history, how they were made, and other details". In: *Frontiers in neuroscience* 9 (2015), p. 481.

[4] G. Orchard et al. "Converting static image datasets to spiking neuromorphic datasets using saccades". In: *Frontiers in neuroscience* 9 (2015), p. 437.

[5] A. Amir et al. "A Low Power, Fully Event-Based Gesture Recognition System". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[6] G. Gallego et al. "Event-based vision: A survey". In: *arXiv preprint arXiv:1904.08405* (2019).

[7] K. A. Boahen. "Point-to-point connectivity between neuromorphic chips using address events". In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 47.5 (2000), pp. 416–434.

[8] R. Benosman et al. "Event-based visual flow". In: *IEEE transactions on neural networks and learning systems* 25.2 (2013), pp. 407–417.

[9] S. Tschechne et al. "Bio-inspired optic flow from event-based neuromorphic sensor input". In: *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*. Springer. 2014, pp. 171–182.

[10] P. Bardow et al. "Simultaneous optical flow and intensity estimation from an event camera". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 884–892.

[11] J. Hidalgo-Carrió et al. "Learning Monocular Dense Depth from Events". In: *arXiv preprint arXiv:2010.08350* (2020).

[12] M. Osswald et al. "A spiking neural network model of 3D perception for event-based neuromorphic stereo vision systems". In: *Scientific reports* 7.1 (2017), pp. 1–12.

[13] A. Z. Zhu et al. "Realtime time synchronized event-based stereo". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 433–447.

[14] G. Gallego et al. "Event-based, 6-DOF camera tracking from photometric depth maps". In: *IEEE transactions on pattern analysis and machine intelligence* 40.10 (2017), pp. 2402–2412.

[15] H. Kim et al. "Real-time 3D reconstruction and 6-DoF tracking with an event camera". In: *European Conference on Computer Vision*. Springer. 2016, pp. 349–364.

[16] G. Orchard et al. "HFirst: a temporal approach to object recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 37.10 (2015), pp. 2028–2040.

[17] J. A. Pérez-Carrasco et al. "Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward ConvNets". In: *IEEE transactions on pattern analysis and machine intelligence* 35.11 (2013), pp. 2706–2719.

[18] G. Lenz et al. *Tonic: event-based datasets and transformations.* Version 0.4.0. Documentation available under https://tonic.readthedocs.io. July 2021.

[19] A. Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *arXiv preprint arXiv:1912.01703* (2019).

[20] Y. LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[21] D. Neil et al. "Effective sensor fusion with event-based sensors and deep network architectures". In: *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2016, pp. 2282–2285.

[22] A. Patino-Saucedo et al. "Event-driven implementation of deep spiking convolutional neural networks for supervised classification using the SpiNNaker neuromorphic platform". In: *Neural Networks* 121 (2020), pp. 319–328.

[23] S. B. Shrestha et al. "Slayer: Spike layer error reassignment in time". In: *Advances in neural information processing systems* 31 (2018).

[24] J. H. Lee et al. "Training deep spiking neural networks using backpropagation". In: *Frontiers in neuroscience* 10 (2016), p. 508.

[25] Y. Wu et al. "Spatio-temporal backpropagation for training high-performance spiking neural networks". In: *Frontiers in neuroscience* 12 (2018), p. 331.

[26] A. Yousefzadeh et al. "Active perception with dynamic vision sensors. Minimum saccades with optimum recognition". In: *IEEE transactions on biomedical circuits and systems* 12.4 (2018), pp. 927–939.

[27] J. C. Thiele et al. "Event-based, timescale invariant unsupervised online deep learning with STDP". In: *Frontiers in computational neuroscience* 12 (2018), p. 46.

[28] G. Giannone et al. "Real-time Classification from Short Event-Camera Streams using Input-filtering Neural ODEs". In: *arXiv preprint arXiv:2004.03156* (2020).

[29] S. Zhou et al. "A Spike Learning System for Event-driven Object Recognition". In: *arXiv preprint arXiv:2101.08850* (2021).

[30] T. Zhang et al. "Self-backpropagation of synaptic modifications elevates the efficiency of spiking and artificial neural networks". In: *Science Advances* 7.43 (2021), eabh0146.

[31] A. Samadzadeh et al. "Convolutional Spiking Neural Networks for Spatio-Temporal Feature Extraction". In: *CoRR* abs/2003.12346 (2020).

[32] A. Safa et al. "Learning Event-based Spatio-Temporal Feature Descriptors via Local Synaptic Plasticity: A Biologically-realistic Perspective of Computer Vision". In: *CoRR* abs/2111.00791 (2021).

[33] B. Yin et al. "Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks". In: *Nature Machine Intelligence* 3.10 (2021), pp. 905–913.

[34] A. Sironi et al. "HATS: Histograms of averaged time surfaces for robust event-based object classification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1731–1740.

[35] L. U. Perrinet. "Feature detection using spikes : the greedy approach". In: *Journal of Physiology-Paris* 98.4-6 (July 2004), pp. 530–9.

[36] L. U. Perrinet et al. "Emergence of filters from natural scenes in a sparse spike coding scheme". In: *Neurocomputing* 58–60.C (2003), pp. 821–6.

[37] P. U. Diehl et al. "Unsupervised learning of digit recognition using spike-timing-dependent plasticity". In: *Frontiers in computational neuroscience* 9 (2015), p. 99.

[38] Y. Wu et al. "Direct training for spiking neural networks: Faster, larger, better". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 1311–1318.

[39] S. Ioffe et al. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. Mar. 2, 2015. URL: http://arxiv.org/abs/1502.03167 (visited on 03/22/2021).

[40] L. U. Perrinet. "Role of homeostasis in learning sparse representations". In: *Neural Computation* 22.7 (July 17, 2010), pp. 1812–36.

[41] L. U. Perrinet. "An adaptive homeostatic algorithm for the unsupervised learning of visual features". In: *Vision* 3.3 (2019), p. 47.

[42] H. B. Barlow et al. "Possible principles underlying the transformation of sensory messages". In: *Sensory communication* 1.01 (1961).

[43] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (Nov. 1998), pp. 2278–2324.

[44] P. Berens et al. "A fast and simple population code for orientation in primate V1". In: *J Neur* 32.31 (2012).

[45] B. Rueckauer et al. "Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification". In: *Frontiers in Neuroscience* 11 (2017).

[46] W. Gerstner. "Time Structure of the Activity in Neural Network Models". In: *Physical Review E* 51.1 (Jan. 1, 1995), pp. 738–758.

[47] B. Nessler et al. "Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity". In: *PLoS Comput Biol* 9.4 (2013), e1003037.

[48] K. Roy et al. "Towards spike-based machine intelligence with neuromorphic computing". In: *Nature* 575.7784 (2019), pp. 607–617.

[49] C. Frenkel et al. "A 28-nm Convolutional Neuromorphic Processor Enabling Online Learning with Spike-Based Retinas". In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2020, pp. 1–5.

[50] P. Lennie. "Single units and visual cortical organization". In: *Perception* 27.8 (1998), pp. 889–935.

[51] J. D. Murray et al. "A hierarchy of intrinsic timescales across primate cortex". In: *Nature neuroscience* 17.12 (2014), pp. 1661–1663.

Antoine Grimaldi is a PhD candidate in computational neuroscience at the Institut de Neuroscience de la Timone in Marseille, France. He obtained a MSc in cognitive science at Phelma, Grenoble-INP, France, and developed an interest in visual perception through the study of natural image statistics and neural representations. His current project focuses on the dynamics of visual information processing using event-based cameras and spiking neural networks. He aims at mimicking some biological mechanisms that govern the emergence of cortical self-organization.

Victor Boutin is a Postdoctoral Research Associate at the Artificial and Natural Intelligence Toulouse Institute (ANITI). He received a Ph.D. in computational neuroscience and artificial intelligence from Aix-Marseille University (France) in 2020 and an MSc in Statistics and Applied Mathematics from Ecole Centrale (France) in 2011. His research seeks to understand how the visual cortex can learn useful representations to generalize visual concepts from a small amount of labeled data.

Sio-Hoi Ieng is currently an Associate Professor with Sorbonne Université, Paris, France, and a member of the Vision Institute, Paris. He was involved in the geometric modeling of catadioptric and non-central vision sensors. His current research interests include neuromorphic and event-based vision perception algorithms and computer vision, with a special reference to the understanding of general visual sensors, exploring cameras networks, and studying the connection between biologic and artificial vision.

Ryad Benosman is a professor at the University of Pittsburgh and Carnegie Mellon University. He is a pioneer of event based sensing and computation. His interest focuses on neuromorphic architectures, event-based computation and sensing and their applications to Bioengineering, Robotics and Space Awareness. His current research interests include the understanding of the computation operated by the visual system with the goal to establish the link between computational and biological vision. He is also invested in applying neuromorphic computation to retina prosthetics and brain decoding, he is the cofounder of several startups based on neuromorphic technologies among them: Prophesee, Pixium Vision, Chronolife.

Laurent Perrinet is a computational neuroscientist currently at the Institut de Neurosciences de la Timone in Marseille, France. His research program is focusing in bridging the complex dynamics of realistic, large scale models of spiking neurons with functional models of low-level vision. In particular, he has developed experimental protocols in collaboration with neurophysiologists to characterize the response of populations of neurons. Recently, he extended models of visual processing in the framework of predictive processing in collaboration with the team of Karl Friston at the University College of London. His current challenge within the NeOpTo team is to translate this mathematical formalism with the event-based nature of neural information, with the aim of pushing forward the frontiers of Artificial Intelligence systems.