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

DOI

10.36227/techrxiv.18003077.v1

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 able to 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. Our first contribution was to add a homeostatic gain control on the activity of neurons to improve the learning of spatio-temporal patterns [2]. A second contribution is to draw an analogy between the HOTS algorithm and Spiking Neural Networks (SNN). Following that analogy, our last contribution is to modify the classification layer and remodel the offline pattern categorization method previously used into an online and event-driven one. 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 and bio-realistic. Results were validated on different datasets: DVS_barrel [3], Poker-DVS [4] and N-MNIST [5]. We foresee to develop the SNN version of the method and to extend this fully event-driven approach to more naturalistic tasks, notably for always-on, ultra-fast object categorization.

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 makes 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 light change 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 such as is defined in the benchmark of digit recognition defined by the event-based, augmented version of MNIST called N-MNIST [5]. Some related approaches have used standard Artificial Neural Networks (ANN) which were converted to Spiking Neural Networks (SNN), resulting

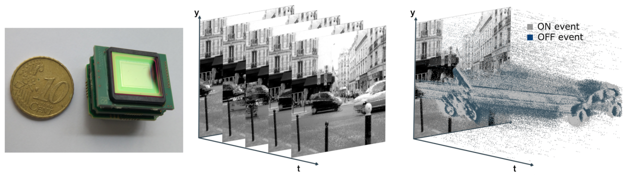

Fig. 1: A miniature event-based ATIS sensor (Left) which, compared to classical frame-based representations (Middle), outputs an event-based representation of the scene (Right).

in overall good classification results [16–18]. Alternatively, some other competitive event-driven algorithms are developed using backpropagation adapted for SNN [19–21]. More recently, it was proposed to introduce biomimetic saccades to boost object recognition [22]. In [1], object recognition is achieved through 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. The algorithm was shown to make accurate predictions on a letter and digit dataset [3], on a flipped card dataset [23] and on a dataset of scenes with faces.

We identified two main limitations in the HOTS algorithm. While the algorithm is efficient for some datasets, we first observed a performance drop when learning to classify digits from the more challenging N-MNIST dataset. In particular, an unequal activation frequency of the different features during learning can lead to a poor variety of time surfaces and consequently to a loss in efficiency. In a subsequent work from the same group [24], this problem was circumvented by averaging time surfaces gathered in a temporal window Δt. To directly address this problem, 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 input events stream, 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 an histogram of the neural activations in the final layer of the network to 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. Even if one can argue that the dynamics of the event-based recording are already captured by the structure 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. Thanks to the increase in robustness offered by the homeostatic gain control, we will here include and test an online classification algorithm at the last layer of the network, such that we hope to achieve a fully end-to-end event-driven and online pattern categorization.

In that perspective, this paper is organized as follows. First, we will present the HOTS algorithm using a novel mathematical formalization, along with the contribution brought by homeostasis. Then, we will extend the categorization algorithm by including a simple bio-plausible online classification layer. Moreover, we will demonstrate that our method corresponds to a SNN with leaky integrate-and-fire (LIF) neuron models. This will allow us to present this unified theoretical framework as a novel contribution as it establishes a bridge between neuromorphic engineering methods used here and computational neurosciences. We have simulated the model for different datasets and a full implementation of this algorithm is available at https://github.com/SpikeAI/HOTS. These scripts allow to reproduce all results presented in this paper and we will give links to reproducible notebooks within the text. Finally, we will show the quantitative improvements of the resulting classification performances, and how its dynamics may vary for different datasets.

2 Materials and Methods

In this section, we start by describing the datasets used in this study and present a method to test our algorithm's robustness to spatial and temporal jitter. Then, we generalize the event-based HOTS model, already described in [1], and extend its formalism to the continuous time domain. After introducing the homeostasis regulation rule that allows for a better learning of the weights of the different layers [2], we describe a new classifier using Multinomial Logistic Regression (MLR) to propose an end-to-end event-driven classification algorithm.

2.1 Datasets

To load the events, we use the community-built *tonic* python package [25]. It currently offers the possibility to load five different event-based datasets and is based on the PyTorch language [26]. 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 the x and y positions on the pixel grid, the time and polarity values, respectively. Timestamps are given in microseconds and polarities are 0 and 1 respectively for OFF and ON events.

2.1.1 DVS_barrel

The DVS_barrel dataset [3] is a collection of digits and letters captured by a DVS. Movement, and thus events, are created by a rotating barrel on which digits or characters are printed. The dataset is composed of 76 samples divided into 36 samples for the training set (one per class) and 40 samples for the testing set. Each sample of DVS_barrel is a 32×32 pixels event-based recording. The average recording time for the samples of the dataset is 60.8 ms Unlike other datasets, DVS_barrel is not included in the *tonic* package.

2.1.2 Poker-DVS

Poker-DVS [4] is one of the first publicly available DVS recordings from a real-world scene and was used to test performances of [1]. It consists of 131 occurrences of the four different symbols of playing poker cards (clubs, diamonds, hearts and spades). They were extracted from 3 separate DVS recordings while very quickly browsing the cards. The sensor size is 31×31 pixels and recordings last on average 17 ms. In *tonic*, there is an available training set of 48 samples and a testing set of 20 samples.

2.1.3 N-MNIST

To test our model with a more complex dataset, we choose the widely used N-MNIST dataset [5]. This dataset was recorded while moving an event-based camera in front of a screen on which digitalized MNIST digits [27] were projected. MNIST digits are originally 28×28 pixels and they were resized to project on the 28×28 pixel grid of an *Asynchronous Time-Based Image Sensor* (ATIS) camera [28]. For N-MNIST, *tonic* registered maximum values of x and y equal to 34, due to the saccades of the camera, which are kept in this study giving a sensor size of 34×34 pixels.

2.1.4 Data augmentation

To test for the robustness of the proposed algorithm, we also used the *tonic* package to transform and augment the N-MNIST dataset. 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 spikes, we can assume that classification performance should get worse as the jitter increases. Therefore, we will use this module to test differentially the robustness of the algorithm by progressively adding some noise to the input signal. To do so, we use a subset composed of randomly selected samples from the testing set of N-MNIST.

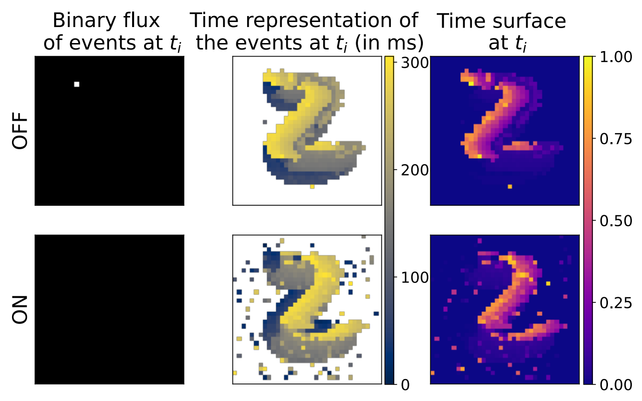

Fig. 2: Illustration of the different event-based data types used in the HOTS network at a given event time. The two rows correspond to the OFF and ON polarities of the events as output of the event-based camera. (Left) Screenshot of one single event (in white) at t_i. (Middle) Timings since the latest 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).

This subset is balanced between the different classes such that we have an equal number of samples for each class. We keep the exact same subset and apply different amounts of spatial and temporal jitter independently. To account for the variability of the jitter applied, we repeat the prediction 10 times for each amount of jitter and derive a statistical quantification from these repetitions.

2.2 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 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* in a similar fashion as Convolutional Neural Networks (CNNs). Let's now formalize these aspects independently.

2.2.1 *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. The rank i is the index of the event and spans from 0 to N_{ev} (excluded). 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 $\mathcal{D}$ of all possible addresses. 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 ($N_p = 2$ for ON and OFF polarities). Each event is usually associated with a time t_i. Note that a stream of events is ordered in time and thus that if $i_1 < i_2$, then $t_{i_1} < t_{i_2}$. 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 instance, $\xi_a(t_i)$ gives the set of events' ranks on address a until the event that occurred at t_i (in particular, including i if it occurred at $a = a_i$).

For the corresponding stream of events occurring at address a, it is possible to construct what we call 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):

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

In particular, since at event time t_i the address a_i is active, it follows that $T_{a_i}(t_i) = t_i$. 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:

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

where τ is a given time constant. This defines a analog vector over the address space that we call the *time surface* and that we write $S(t)$. The value for each given address represents a scalar which is close to 1 when an event recently occurred, and then vanishes to 0 when no event was recently observed. 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.

2.2.2 *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 $\mathcal{A}$ and we may index these 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 to 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

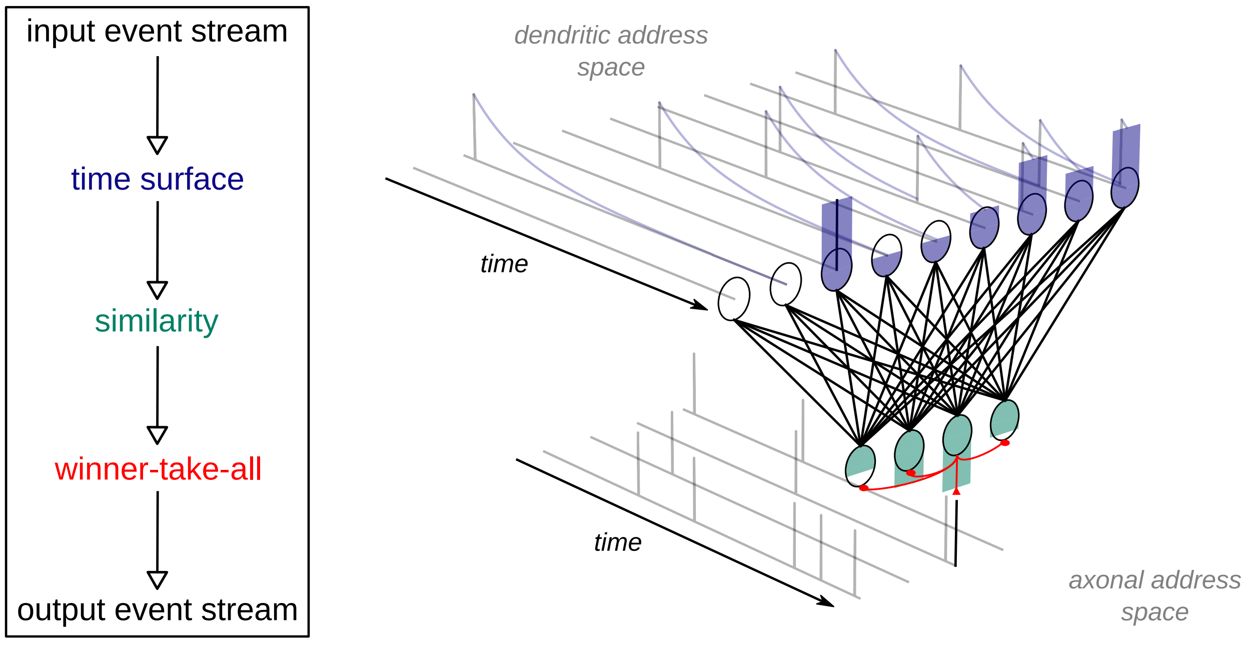

Fig. 3: Illustration of the core computation made within one layer of the HOTS algorithm. On the top of the plot, we show the dendritic stream of events convolved by an exponential decay which forms the time surface. Time surfaces are computed at the timestamp of each event/spike. The time surface at 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.

the similarity of weight patterns with each time surface [29]. The similarity measure $\beta_\mathbf{n}$ is defined as the scalar product over the dendritic space $\mathcal{D}$:

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

Note that the similarity vector over all neurons may be quickly computed as a matrix multiplication using $\beta(t) = W^T S(t)$ (where T denotes the transpose operator). For each incoming event at time t_i, this similarity measure is updated and forms a vector $[\beta_\mathbf{n}(t_i)]_{\mathbf{n}\in\mathcal{A}}$ across all neurons.

Whenever a new event inputs 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 is the most similar to the analog input:

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

Note that this choice depends on the past events history by way of the time surface as these are defined relative to the time context which records (at t_i) the time of the last observed events. 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 $\mathbf{L}$ with, by convention, $\mathbf{L} = 0$ designing the input layer. To describe the input of 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.

2.2.3 *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 operated 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. One notable advantage of this representation lies in the translation-invariance which is imposed in the representation by the convolution operator. 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.

In all generality, the convolution that is processed implies is a linear operation which can thus be modeled as a matrix multiplication. The corresponding weight matrix may be constructed using a Toeplitz operation which replicates the kernel of each channel at all different positions. More formally, let's define the dendritic address space for each layer as $\mathcal{D}^{\mathbf{L}} = [0, N_X) \times [0, N_Y) \times [0, N_p^{\mathbf{L}}) \subset \mathbb{N}^3$. The number $N_p^{\mathbf{L}}$ defines the number of channels of the time surface as input to layer $\mathbf{L}$, we name dentritic them channels. Each address may then 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 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 to compute 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 be computed by a convolution over $\mathcal{A}^{\mathbf{L}}$. In our formalism, time surfaces are defined globally and each weight vector corresponds to one column of the weight matrix which index is associated to an 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 are 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, that is, 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 kernels $N_n^{\mathbf{L}}$, 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.

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]. That is, once a neuron is matched, a Hebbian-like mechanism is used to take the selected weight vector $W_{\mathbf{n}}$ closer to the observed time surface. This mechanism is similar in principle to that used by the k-means algorithm and is implemented in numerous other unsupervised learning schemes [30]. Figure 4 offers an illustration of the different kernels learned by the network.

2.3 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 [31, 32] to balance the firing rate over the neurons of each layer of the SNN. The model of [31] uses an adaptive membrane threshold while [32] 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 [33] used for training deep neural networks. These methods are similar in their objectives and are well justified in terms of efficient coding [34].

Here, we implement homeostasis regulation by adapting the heuristics from a previous work [35]. It simply consists in modifying the similarity measure (see equation (3)) as:

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

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

$$\gamma_{\mathbf{n}}(t) = e^{\lambda \cdot (f(t) \cdot N_n - 1)} \tag{6}$$

where λ is a regularization parameter, f is the relative activation frequency of neuron $\mathbf{n}$ and N_n the total number of neurons in the layer. The homeostatic gain rule is such that $\gamma_{\mathbf{n}} > 1$ if neuron $\mathbf{n}$ is less activated than the *a priori* average $\frac{1}{N_n}$ frequency on the layer and $\gamma_{\mathbf{n}} < 1$ in the opposite case. The activation frequency of the neuron is measured by computing its probability of being activated over all the previous events. This regulation rule allows to train the different neurons in a balanced fashion and avoid the response of only a few of them. Note in particular that when we reach homeostasis and all neurons fire with equal prior probability, then $\gamma_{\mathbf{n}} = 1$ and the processing is similar to that of HOTS. Note finally that when this happens within a layer, the distribution of activation probability is uniform (with probability $p = \frac{1}{N_n}$) and the event-based code reaches a maximum of its entropy: the average information carried by the address of an event approaches the maximal value of $\log_2(\frac{1}{p}) = \log_2(N_n)$ bits.

In practice, we observed that adding homeostasis to balance the activation frequency of neurons leads to a better clustering of the weight matrices, see figure 4. Note that neural activity is balanced during the unsupervised

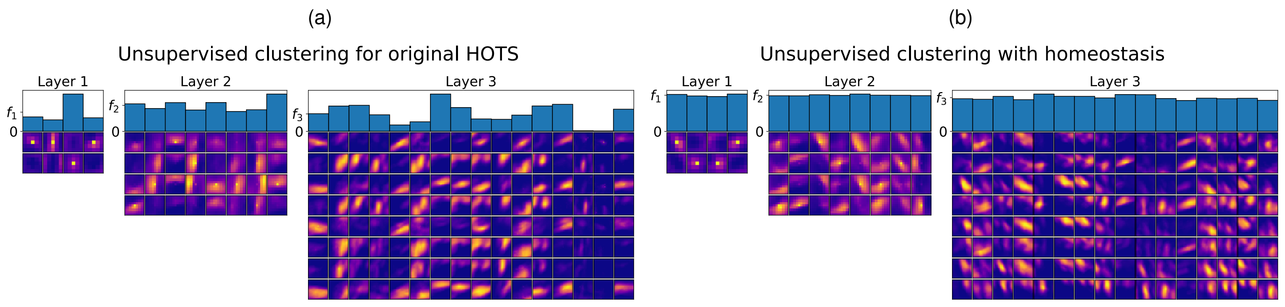

Fig. 4: Activation histograms and time surfaces obtained in the unsupervised learning algorithm (a) for the original HOTS network (replicated from [1]) and (b) for the bio-plausible version with homeostasis. Activation histograms correspond to the frequency by which each neuron was activated. For each 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.

clustering phase across all digits. The homeostasis process does not guarantee an equiprobability of neural activity for one digit, but over the whole learning set in line with the efficient coding hypothesis [36]. In addition, it allows avoiding to introduce an *ad hoc* heuristics in the learning rule (see above) 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. This 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. This method for the unsupervised learning phase makes our algorithm more similar to the conditions faced by living systems.

2.4 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. Online inference on event-based data was developed in past studies [37–39]. However, [39] accumulates spikes as input to reconstruct an image frame, [38] uses a sample-and-hold approach, and freezes events during a defined time step. [37] proposes to use Spike-Timing Dependent Plasticity (STDP) for unsupervised learning of spatio-temporal features. For this last study, they also make 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.

We propose a novel method to perform online event-driven classification. 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 mean duration of the samples. The time surface thus forms an analog vector that may be used in a MLR model to achieve supervised classification. Such MLR models are for instance used in the last layer of classical deep-learning networks [40] and are compatible with a neural implementation [41]. 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:

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

where $W_j^{\mathbf{C}}$ are the coefficients associated to class j of the MLR model. As in section 2.2.2, 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 with the following equation:

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

Where $\beta_c^{\mathbf{C}}(t)$ 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 class c. The final prediction is made 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:

$$c(t) = \underset{c \in \{1,\ldots,N_{\text{class}}\}}{\arg\max} \ \beta_c^{\mathbf{C}}(t)$$

It results in an always-on decision process able to make a prediction at any time. Here, we 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.

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. 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. This allowed us to compute an accuracy on an event-by-event basis.

3 Results

3.1 The Spiking Neural Network analogy

We have defined the HOTS algorithm in an event-based formalism and our first result is to demonstrate that when it is extended to the continuous-time domain, this algorithm may be implemented as a Spiking Neural Network (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. 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 computational neuroscience framework.

3.1.1 *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):

$$\forall t \in \mathbb{R}^+, \quad S_a(t) = e^{-\frac{t - T_a(t)}{\tau}}$$

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 [42] 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:

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

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 only the time until the last spike has an influence and not spikes before that, as implemented in the definition of the time context. As a consequence, it implements a form of resetting mechanism which allows to compute 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:

$$\frac{\mathrm{d}}{\mathrm{d}t} \beta_{\mathbf{n}}(t) = -\frac{1}{\tau} \cdot \beta_{\mathbf{n}}(t) + \sum_{i=0}^{N_{ev}} w_{\mathbf{n},a_i} \cdot (1 - S_{a_i}(t)) \cdot \delta(t - t_i) \quad (9)$$

Such a ODE is classical for the description of the evolution of the membrane potential of Integrate-and-Fire neurons. Note that the major change lies in the modulation of the integration of incoming spikes which allows to represent only the time until the last spike. The hierarchical network proposed in [1] is then equivalent to a SNN of LIF models with a Hebbian-like learning mechanism as mentioned in section 2.2.3. 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. This will generate a spike with the same characteristics as the input event (position, time). Only the channel, or polarity, will change, and is given by the index of the winning spiking neuron. In the hierarchical network that we defined, the number of neurons will typically increase in each layer. As a consequence, we identify that there is an inconsistency in the HOTS model that we use here with biological neurons through the lack of a fixed potential threshold (leading to the absence of response latency). This delay is a constraint in the biological mechanism for spike emission and in this framework, we argue that we could draw a biological analogy while being faithful to the original event-based algorithm. One advantage of this method over even more realistic networks is the absence of delay on the computations that allows for fast classification.

3.1.2 *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 [41]. Analogous to biology and as described in section 3.1.1, 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 (7) is a good model of a spiking WTA network. Indeed, [43] 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 θ_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 and learning of the weights, done in an event-driven fashion, corresponds to a Hebbian-like mechanism for the neurons inside the network and in the classification layer.

3.2 Replication of HOTS and the role of homeostasis

In a recent paper [2], we reproduced results found in HOTS [1] with the DVS_barrel and Poker-DVS datasets. Results are accessible online at 01 SimpleAlphabet and 02_PokerDVS. We replicated past results with the first dataset and, with the Poker-DVS dataset, we demonstrated the inconstancy in classification performance emerging from the variability of the unsupervised clustering in the original method [1]. This variability in the clustering phase is mainly due to the initialization of the synaptic weights with the first time surfaces entering the network, a common problem of unsupervised learning algorithms such as k-means.

To circumvent that issue, we initialized matrices at random and applied a homeostatic gain control mechanism over neural activations, resulting in a more stable learning of the synaptic weights, see figure 4. Results were compared for both methods, with and without homeostasis regulation, reporting an improvement when adding homeostasis. The classification task was also performed on the widely used N-MNIST dataset with, again, improved performances for the method with the homeostatic gain control mechanism. One result of interest is that homeostasis significantly improved robustness to both spatial and temporal jitter. In this paper, we make use of the advantages brought by homeostasis, and we present a new classifier allowing online classification to further extend these results.

3.3 Online inference

In this section, we present the results for the end-to-end event-driven online classification described in section 2.4. First, to illustrate the dynamic evolution of classification performances at the event scale, we will plot the accuracy value for each event on the data stream since its beginning. Within a dataset, the total number of events for the samples can vary. We set a maximum number of events to compute accuracy by taking the $95th$ percentile of the dataset in terms of number of events. We also report the classification performance as a function of time by computing the accuracy value at defined time steps. Within each time bin, we set the predicted class as the one that was chosen most by the classifier. As recordings within the same dataset have approximately a fixed duration, we report the mean accuracy value on a timescale defined by the duration of the shortest sample. Finally, to compare with other methods, we report classification performance by making one decision per sample based on the class that was chosen the most, we call this measure offline accuracy. Moreover, we compare the classification performances with MLR on the raw event-based signal and on the output of the network in order to quantitatively assess the coding efficiency of the trained network. This is denoted as *online RAW* in the plots. We also name *original HOTS*, results replicated with the method described in [1] and *online hHOTS* the event-driven algorithm described in our study.

3.3.1 *DVS_barrel*

We start by the DVS_barrel dataset, which is composed of few samples (36 for training and 40 for testing), and serves as a useful toy model. We reproduced past offline classification results obtained in [1] and then observed the online classification performances. For the analog vector we feed to the classifier, we used a time constant $\tau_{\mathbf{L}=c} = 50$ ms. In figure 5, one can see the previous classification performance for the original method with histogram comparison, represented as a red dashed line. In this latter methods, the prediction of the class can only be done offline, once the classifier received all the events. The accuracy value obtained for this method is close to 98 %. We computed the offline accuracy of the other methods and obtained 100% for both online RAW and online hHOTS.

Figure 5(a) illustrates the evolution of the accuracy as a function of the number of events treated by the classifier. One can notice that this accuracy value is outperformed by the online method (in blue) after only a few number of events or a reduced amount of time compared to the full recording. Surprisingly, the method that performs classification with a single MLR model on the raw event stream (in orange), that is the output of the event-based camera,

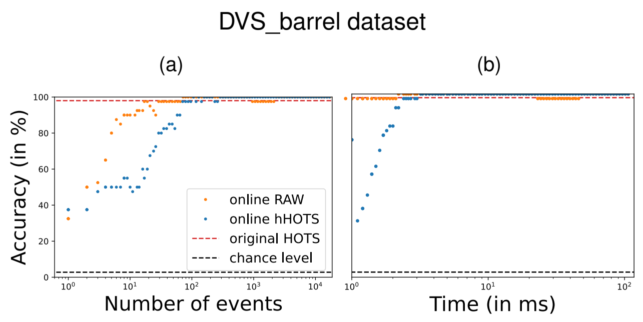

Fig. 5: Online classification performance on the DVS_barrel dataset. We show the average accuracy computed with respect to the number of events since the beginning of the stream (a) and also as a function of the corresponding time (b).

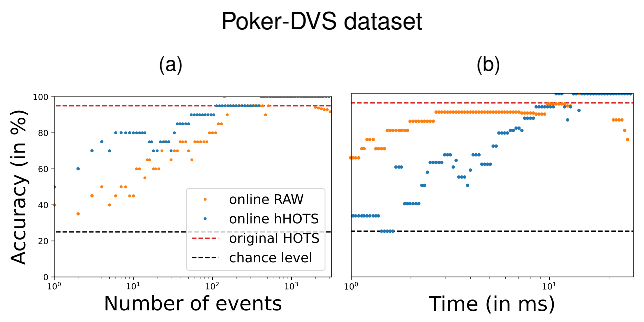

Fig. 6: Online classification performance on the Poker-DVS dataset. We show the average accuracy computed with respect to the number of events since the beginning of the stream (a) and also as a function of the corresponding time (b).

also outperforms the past method after few events. For the first events, there is an advantage for the online method performing MLR on the raw stream of events. Then, both online methods outperform the original HOTS algorithm with only $\approx$ 80 events. After $\approx$ 250 events, our method obtains an accuracy of 100 % and the classifier using the raw event stream as well, with a small drop in performance around 1000 events. Even if it seems to be more stable after 250 events, there is no significant advantage for the method that uses the output of the HOTS network. This surprising result may originate from the fact that samples are all presented in a rotating barrel. The DVS is fixed and the movement of the digits is close to a uniform translation. As such, the spatio-temporal signature of the digits will be easy to capture and the MLR model alone is enough to make a good prediction.

In figure 5(b), the accuracy value is presented as a function of time. The time step used for this dataset is 100 μs and, as explained before, we make the prediction by choosing the class that is selected the most during a time step. The online RAW method shows excellent results and reaches the same performances of the original method at the second time step, that is, after only 1 ms. The method using the output of the HOTS network to feed the classifier performs reaches 98% of accuracy only at 2.5 ms. We explain this delay to reach good classification results by the filtering of the events done within the network. Indeed, each time surface that is sent to a layer of the network needs a minimum number of non-zero components to be treated and trigger a new event. As a consequence, the network needs to gather a certain amount of events to send the first spike to the classifier, leading to a time delay for the first predictions and for the accumulation of evidences. Notice that both online methods reach 100% accuracy after $\approx$ 4 ms. Once again, we observe that only the very beginning of the recording is sufficient to reach a perfect recognition.

3.3.2 Poker-DVS

For the Poker-DVS dataset, we used a time constant for the classification layer equal to 10 ms. This value is significantly smaller than for the other datasets, as the recordings in this dataset are very short, with an average of 17.4 ms. Previous results obtained in [2] gave an accuracy of 95 % for the HOTS algorithm (see 02_PokerDVS for more details). We obtain 100% for the offline accuracy of both methods using the MLR model as a classifier.

Figure 6(a) shows the evolution of the classification performance as a function of the number of events processed by the classification layer. One can see that, for our online method, accuracy increases rapidly with a few events and reaches a plateau at 100 % of accuracy after about 500 events. For the raw event stream, MLR also performs well, even if it reaches the maximum accuracy plateau slightly later. Both methods match the original classification results after about 200 events. Note that, classification on the raw inputs is less stable at the end of the recording and drops in accuracy for more than 2500 events.

Figure 6(b) gives the classification accuracy relative to the time of the recording. Time steps used for classification are of 10 μs each. As in figure 5(b), we observe a delay for the first predictions of our method. MLR on the raw event stream can make prediction for the very first events, but shows worse accuracy values at the end of the recording. For these very short recordings, perfect recognition is obtained, only for our method, at the end of the event stream, after around $\approx$ 15 ms. Note that the decision of the class for one specific time step gathers all the predictions taken for each event with this time step. This difference to choose the predicted class explains the non-linear changes on classification performance when going from the events scale to the time scale.

3.3.3 N-MNIST

We finally test our method on a relatively more complex task, the N-MNIST dataset. We first check for the results with the original method from [1] and see that performances are 37.9 % of accuracy on the N-MNIST dataset. This classification score is computed when using one histogram averaged for all the samples in the same class. In a previous study, authors used a k-Nearest Neighbors to boost performances of the method with a simple classifier, more appropriate for big datasets like N-MNIST [44]. With $k = 12$, we reached an accuracy of 86.9 % for the original HOTS algorithm (see red dashed line in figure 7), and improved

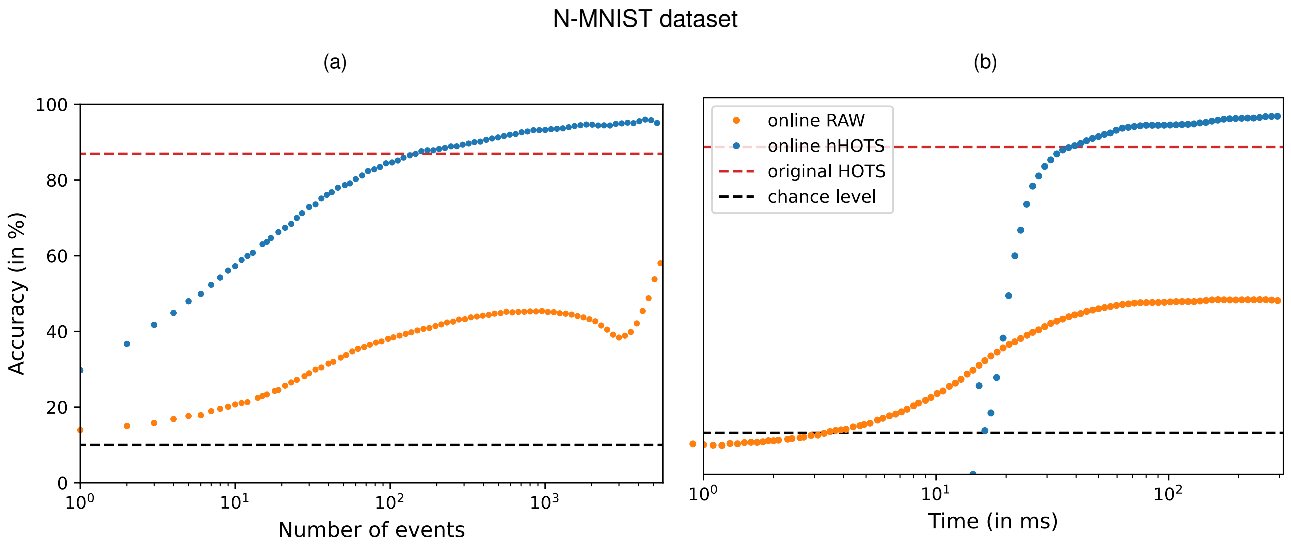

Fig. 7: Online classification performance on the N-MNIST dataset. We show the average accuracy computed with respect to the number of events since the beginning of the stream (a) and also as a function of the corresponding time (b).

its classification performance with homeostatic gain control to reach 87.7 % (not shown here). These values are obtained with histograms gathering the activation distribution of the last layer of the network.

Let's now compare these results with the online classifier. For this dataset, we used $\tau_c = 150$ ms for the time constant of the online classification layer. Also, to reduce the total computation time, the learning of the MLR model on the training set was done with a down sampled stream of events. Only 1 event every 10 was kept, and we will see that it is sufficient for a robust training of the classification layer. After training, we tested the classifier and figure 7 illustrates the evolution of the classification performances during the recordings. One can observe a strong advantage for the method that uses the encoding of HOTS compared to the control method making classification on the raw event stream.

As for previous datasets, we report classification accuracy when making one decision per sample based on the class that was chosen the most, with decisions made at the event scale. For the MLR classifier using the output of the HOTS network, we obtain 95.1 % of accuracy and only 44.6 % for the control method. In this case, the shapes of the digits are more complex and diverse than for previous datasets, giving evidence that encoding by the hierarchical network is here essential to perform a correct classification. Note that higher classification results obtained on the N-MNIST dataset are reported in the literature. [32] announces 99.4 % of accuracy for a 8-layered SNN trained with back-propagation. Different works using back-propagation obtain very high accuracy: ≈ 99 % [20, 45, 19, 46]. We argue that, even if we don't outperform these state-of-the-art results, this simpler feedforward network structure with a bio-plausible learning obtains very competitive accuracy values.

Figure 7(a) gives accuracy values as a function of the number of events used for classification. Unsurprisingly, classification accuracy improves as the number of events increases. Quickly after ≈ 150 events, our method (in blue) outperforms the offline classification method used in [2] and the mean accuracy keeps increasing with the number of events. This online classification allows ultra-fast object categorization in terms of events: only with a few events, classification reaches a good level of accuracy. The control method, on the other hand, stays below classification performances obtained with the original HOTS. The non-stable behavior observed for the orange curve after 1000 events is due to the variability in the number of events in each recording.

In figure 7(b), we observe the evolution of classification accuracy as a function of time. Time steps used for this dataset last 100 μs and classification is done by making a decision per time step based on the class that was matched the most by the softmax function. This difference in categorization, made on a defined time step and not for every event, can explain the drop in accuracy for the first decisions for both online methods. Indeed, the first accuracy values remain lower than in 7(a), that is when the classifier makes a decision per event. Then, when enough events are gathered, accuracy values are stabilizing and match the classification results obtained as a function of the number of events. For our method, the first decisions, which correspond to the first events emitted as the output of the HOTS network, are being performed after approximately 10 ms. Then, classification accuracy increases rapidly. Our method outperforms the original one after ≈ 30 ms and then keeps increasing to reach a plateau. N-MNIST recordings are composed of three saccades lasting 100 ms each. The performance of the classification at the end of the first saccade is 92.7 % of accuracy, at the end of the second saccade we reach 94.5 % of correct categorization and at the end of the third saccade we reach 95.1 % of accuracy. Note that, already at the first saccade, our method is able to obtain good classification results. The two remaining saccades improve categorization performances to reach the accuracy value

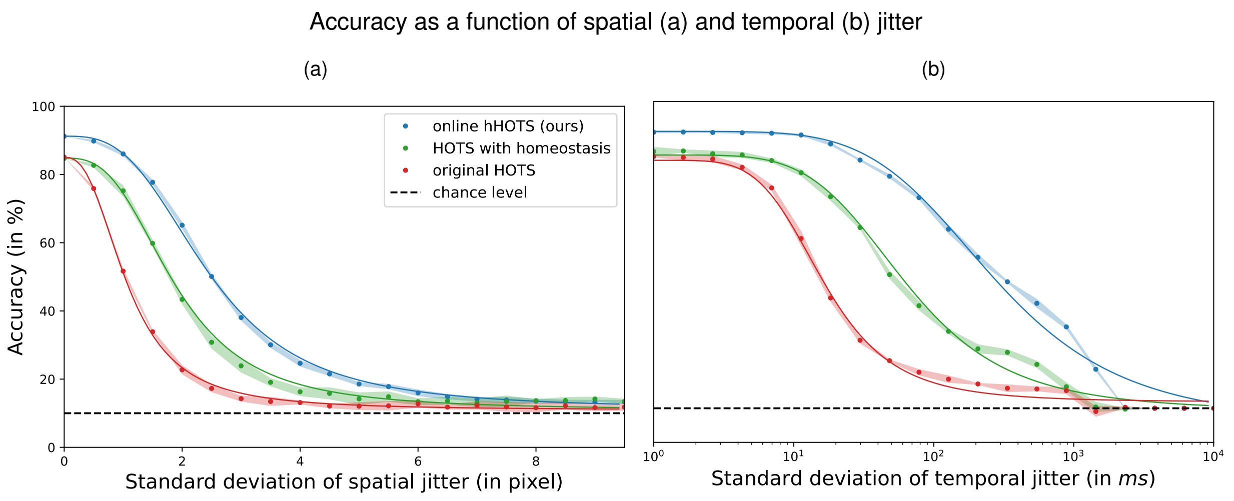

Fig. 8: Robustness of classification performances on N-MNIST. The input stream was perturbed by changing the position or timing of the events and we report classification results as a function of (a) standard deviation of temporal jitter in logscale or (b) standard deviation of spatial jitter. Blue curves show fits for the results of the HOTS method with homeostasis and these are compared to the performance of the original HOTS algorithm (Red curves). Dots are the mean values of accuracy over 10 trials for discrete values of jitter. Transparent outlines represent the 5% and 95% quantiles.

obtained when taking one decision per sample. Both as a function of the number of events or as a function of jitter, our method allows reaching good classification results with only a small portion of the total recording. This algorithm shows interesting performances for ultra-fast digit recognition. The results of 7(b) indicate that the second and the third saccade of the N-MNIST recordings add only a small amount of information. Previous works report accuracy results using only the first saccade and show only small improvement when using the other ones as well [20, 18, 37].

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. To assess the performance of the method, a subset of 1000 samples of N-MNIST is taken randomly and used to apply different amounts of jitter. 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 these results to compute the percentiles plotted in figure 8. To compare with past results obtained in [2], we plot the offline accuracy obtained when making one decision per sample. When no jitter is applied, the original HOTS method and the algorithm with homeostasis get comparable results. The online HOTS presented in this study performs better, as reported above. We obtain a significant increase in performance with our online classifier for all the different amounts of jitter. 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 (i.e. $10\ \%$ accuracy). 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.

In figure 8(a), we show the mean accuracy for the original method, the one with homeostasis and ours. Accuracy reaches its half-saturation level for a spatial jitter with a standard deviation equal to 1.35, 2.30 and 2.94 pixels, respectively. Homeostasis increases resilience to spatial jitter when applied to the original method. As shown in [2], there is a qualitative improvement in the clustering phase when applying homeostasis. Indeed, kernels learned by the network with homeostasis were all matured and offered a more diverse dictionary for an efficient encoding by the network. With the online classifier, robustness to jitter is again increased and the half-saturation level is more than doubled compared to the one from the original method.

Figure 8(b) illustrates a high resilience of the network to temporal jitter. The x-axis is composed of log_{10} values, and one can observe that even the original method is robust to temporal jitter. The half-saturation values are $23.44\ \mathrm{ms}$, $100.0\ \mathrm{ms}$ and $407.4\ \mathrm{ms}$ respectively for the original HOTS method, for the one with homeostasis and for the algorithm presented in this study. Even the original method offers a high resilience to temporal jitter. 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 network. For classification with online MLR, half-saturation level is reached for a standard deviation of temporal jitter equal to 407.4 ms. The samples of the N-MNIST dataset last on average 307.7 ms, meaning that the chronological order of the events has a limited impact for the method and task used here. 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, the combination of only few events on precise spatial locations can lead to a good prediction of the class. On this specific dataset of static binary digits moving in front of a DVS, temporal information seems not to be that important for classification. We expect sensibly different qualitative results on more realistic datasets.

4 Discussion

In this study, we augmented a neuromorphic engineering method with techniques from computational neuroscience to develop an online, event-driven classification algorithm. On one hand, neuromorphic engineering is inspired by biology to develop efficient algorithms and technology. For instance, DVS cameras are inspired by the retina, memristors are used to implement neuroplasticity and neuromorphic chips are designed to emulate neural elements in order to perform sparse and asynchronous computations as observed in the biological brain. Therefore, the neuromorphic field is interested in understanding neural circuitry to be able to reproduce such mechanisms. On the other hand, computational neuroscience uses mathematical models to describe and explain a wide range of cognitive processes. It focuses on biological plausibility to validate its models and in comparison with experimental data. As a result, both fields have a common interest in neural information processing and how these may implement the computations that happen in the brain (for a review, see [47]).

We initiated our study with the HOTS network which original basis is inspired by the hierarchy found in the visual cortex. As designed in this network, the receptive field size is increasing along the visual hierarchy [48]. In [49], it is observed that cortical areas follow a hierarchical ordering of intrinsic timescales. They infer that shorter timescales can be useful for rapid detection or tracking of dynamic stimuli, whereas longer timescales can be used for decision-making computations made by higher level areas. This particular organization in the HOTS architecture and the evolution of the parameters of time surfaces across the different layers follows physiological principles. From a computational perspective, time surfaces capture the inter spike interval (ISI) characteristics by being a function of the delay between the current time and the last event on a specific dendritic address. The exponential decay applies a non-linearity to this delay and results in a finer resolution of the analog value for the recent events decreasing rapidly with time to vanish after only few τ. The parameter τ, used to compute time surfaces, is then defined to obtain an optimal representation of the ISI distribution on the different addresses. In the HOTS network, there is an increase of the number of neurons, thus an increase of the number of channels, across the different layers. Using principled theory, one can demonstrate that the increase of the number of channels along the hierarchy will impact the ISI distributions. Indeed, from the core mechanism of HOTS, one event is emitted for each event as input of a layer, then by increasing the number of channels, the distribution of the spikes will get more and more sparse and the ISI will increase. To keep an optimal representation, the time constant of the exponential decay has to be increased as well similarly to what is observed in the choice of the hyper parameters of the HOTS network. In line with theoretical neuroscience, we join the formalism of the event-based algorithm that we propose with SNN computations by extending the equations to the continuous time domain. We also define the core computations of the network on generic address spaces and this new mathematical formalism may seem to complicate the formulation of the computations. We argue that by bridging these two different fields and develop a common framework, one can open interesting cross-talks about advances done in parallel from both sides. In this work, we continue to use computations observed in nature to progress toward more generic and bio-plausible models. First, we used a homeostatic rule inspired by living systems to make the unsupervised online learning of the network more robust. Then, we added an online classification layer that performs MLR and which is compatible with a neural implementation [41]. Globally, the results offer a good example of the possible joint future between neuromorphic engineering and computational neuroscience. In this method, the clustering phase, i.e. when the network learns kernels, is fully online and unsupervised. We have shown in a previous study that this clustering can be very sensitive to initialization [2]. Thanks to the homeostatic gain control that balances the activity of the neurons within the same layer, unsupervised learning can be stabilized. The classification layer learns the weights of the MLR in a supervised way. This learning is also performed online: digits are presented independently, one after the other. The temporal dynamics between the events of the same digits are given by the temporal decay τ_c and learning can be done for each analog vector as input of the classification layer in an online fashion. This online, bio-plausible and event-driven learning method is interesting for applications in systems with a reduced memory like embedded architectures, for instance in autonomous robots. Once trained, the network can realize always-on classification, meaning that, whenever it is needed, a prediction can be inferred. We present results obtained with event-driven categorization, that is a prediction if done for each event as input of the classification layer. It results in ultra-fast categorization because there is no need to wait for the end of the sample recording or to gather a definite amount of events. As we propose a generic formalism, predictions can be done with different constraints, for instance based on specific timings or when the output of the classification layer reach a fixed threshold leading to stronger confidence in the prediction.

This dynamical classification, that evolves through time for each new event, is more similar to object recognition performed by biological systems [50]. For the datasets presented in this paper, only a reduced number of events in the sample were needed to reach good accuracy (see for instance figure 7). We also observed a surprisingly high robustness of the method to temporal jitter. We infer that, for this dataset made from static and binary digits, and because spatial locations are transferred along the different

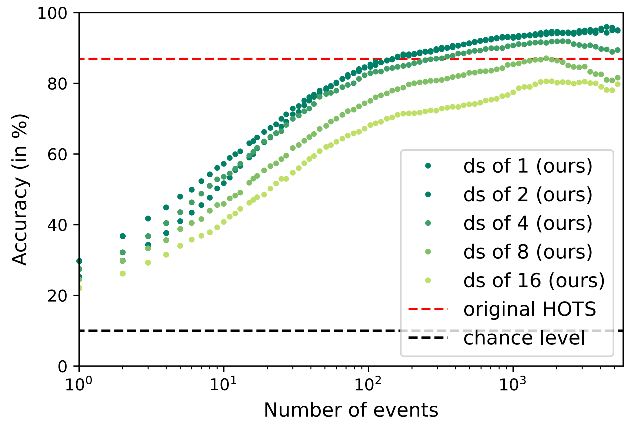

Fig. 9: Effect of the spatial downsampling on online classification accuracy as a function of the number of events on the N-MNIST dataset. MLR with a ds (downsampling) of 1 corresponds to the method presented in this paper and as the downsampling increases, the spatial resolution of the pixel grid decreases. 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 given by the event stream as the output of the network.

layers, the MLR model can capture some specific relative locations of the output spikes to perform good enough decisions. To assess this hypothesis, we applied a spatial downsampling on the global time surface that serves as input of the classifier. The MLR model is then learned and tested with the spatial resolution of the pixel grid, in this case 34×34, reduced progressively by the spatial downsampling. Results are presented in figure 9 and, along with results of figure 8(a), highlight the importance of spatial information in the stream of events as output of the network. In comparison, the classifier does not seem to be sensitive to temporal ordering of the events, but we argue that this may be due to the characteristics of the dataset. We aim at applying this method to real-world recordings in order to assess performances for more complex tasks.

One potential concern when adapting this algorithm to real-world categorization is the use of time surfaces to detect an object. Indeed, with a deeper analysis of the representation inherent to time surfaces, one can see that they represent the dynamical signature of a moving object. A time surface represents therefore a combination of an object and its motion, but both elements are not explicitly dissociated. This entangled information may need to be separated in order to reach optimum object categorization. For simple datasets with a uniform movement like DVS_barrel and Poker-DVS, classification is excellent. For more complex shapes like in N-MNIST, we can still perform good classification accuracy, yet we cannot reach the performances of state-of-the-art algorithms. Even if challenging results are obtained with a reduced number of events, the classification accuracy reaches a plateau, and we think that we can overcome this limitation in performance by dissociating shape from motion in the time surface.

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] 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.

[4] 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.

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

[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] 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.

[17] 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.

[18] 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.

[19] S. B. Shrestha et al. "Slayer: Spike layer error reassignment in time". In: *arXiv preprint arXiv:1810.08646* (2018).

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

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

[22] 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.

[23] 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.

[24] 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.

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

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

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

[28] C. Posch et al. "A QVGA 143dB dynamic range asynchronous address-event PWM dynamic image sensor with lossless pixel-level video compression". In: *2010 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE. 2010, pp. 400–401.

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

[30] 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.

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

[32] 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.

[33] 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).

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

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

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

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

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

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

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

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

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

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

[44] J.-M. Maro et al. "Event-based gesture recognition with dynamic background suppression using smartphone computational capabilities". In: *Frontiers in neuroscience* 14 (2020), p. 275.

[45] G. K. Cohen et al. "Skimming digits: neuromorphic classification of spike-encoded images". In: *Frontiers in neuroscience* 10 (2016), p. 184.

[46] Y. Jin et al. *Hybrid Macro/Micro Level Backpropagation for Training Deep Spiking Neural Networks*. 2019.

[47] F. Zenke et al. "Visualizing a joint future of neuroscience and neuromorphic engineering". In: *Neuron* 109.4 (2021), pp. 571–575.

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

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

[50] S. Thorpe et al. “Speed of processing in the human visual system”. In: *nature* 381.6582 (1996), pp. 520–522.

Antoine Grimaldi 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 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 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 Ryad B. 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 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, as part of the FACETS and BrainScaleS consortia, he has developed experimental protocols in collaboration with neurophysiologists to characterize the response of population 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.