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

Antoine Grimaldi, Victor Boutin, Sio-Hoi Ieng, Ryad Benosman, Laurent U Perrinet

▶ **To cite this version:**

Antoine Grimaldi, Victor Boutin, Sio-Hoi Ieng, Ryad Benosman, Laurent U Perrinet. A robust event-driven approach to always-on object recognition. 2024. hal-04694717

HAL Id: hal-04694717
https://hal.science/hal-04694717v1

Preprint submitted on 11 Sep 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

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

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

[a]*Institut de Neurosciences de la Timone (UMR 7289); Aix Marseille Univ, CNRS; Marseille, France,*

Abstract

We propose a neuromimetic architecture that can perform always-on pattern recognition. To achieve this, we have extended an existing event-based algorithm (Lagorce et al., 2017), which introduced novel spatio-temporal features as a Hierarchy Of Time-Surfaces (HOTS). Built from asynchronous events captured by a neuromorphic camera, these time surfaces allow to encode the local dynamics of a visual scene and to create an efficient event-based pattern recognition architecture. Inspired by neuroscience, we have extended this method to improve its performance. First, we add a homeostatic gain control on the activity of neurons to improve the learning of spatio-temporal patterns (Grimaldi et al., 2021). We also provide a new mathematical formalism that allows an analogy to be drawn between the HOTS algorithm and Spiking Neural Networks (SNN). Following this analogy, we transform the offline pattern categorization method into an online and event-driven layer. This classifier uses the spiking output of the network to define new time surfaces and we then perform the online classification with a neuromimetic implementation of a multinomial logistic regression. These improvements not only consistently increase the performance of the network, but also bring this event-driven pattern recognition algorithm fully online. The results have been validated on different datasets: Poker-DVS (Serrano-Gotarredona and Linares-Barranco, 2015), N-MNIST (Orchard et al., 2015a) and DVS Gesture (Amir et al., 2017). This demonstrates the efficiency of this bio-realistic SNN for ultra-fast object categorization through an event-by-event decision making process.

Keywords: vision, pattern recognition, event-based computations, spiking neural networks, homeostasis, efficient coding, online classification

1. Introduction

Bio-inspired engineering aims to take advantage of our understanding of nature's complex and impressively efficient mechanisms. Event-based cameras perfectly illustrate this process. These sensors, also known as silicon retinas, are inspired by biological retinas and make it possible to capture light information asynchronously. Unlike their classical frame-based counterpart, an event-based camera reacts to the dynamics of the scene on a pixel-by-pixel basis: when a change in luminance is detected, an event is emitted. The event is labelled with an ON or OFF polarity depending on whether it corresponds to an increase or decrease in brightness, respectively (see figure 1). Event-based cameras offer several advantages such as a high temporal resolution, energy efficiency, redundancy reduction and a high dynamic range. They are many interesting applications and use cases for event-based cameras now flourishing in the scientific community (see Gallego et al. (2019) for a review). This new technology, together with the corresponding Address Event Representation specification (Boahen, 2000), brings a paradigm shift in the way visual information is processed. Efficient event-driven solutions have been found to solve classical computer vision tasks such as the optical flow estimation (Benosman et al., 2013; Tschechne et al., 2014; Bardow et al., 2016), 3D reconstruction (Hidalgo-Carrió et al., 2020; Osswald et al., 2017; Zhu et al., 2018) or the simultaneous localization and mapping problem (Gallego et al., 2017; Kim et al., 2016). In this work, we focus on performing pattern recognition and extend an already existing method: Lagorce et al. (2017).

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. These are then combined into 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 that appear in the event stream. This algorithm has been shown to make accurate predictions on a letter and digit dataset (Orchard et al., 2015b), on a flipped card dataset (Pérez-Carrasco et al., 2013) and on a dataset of scenes with faces.

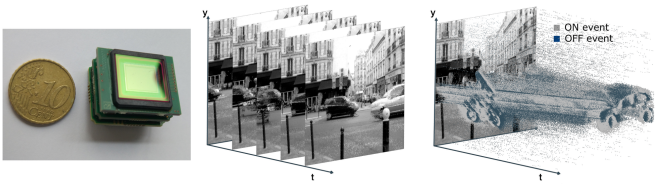

Figure 1: A miniature event-based ATIS sensor (Left) which, compared to classical frame-based representations (Middle), outputs an event-based representation of the scene (Right).

We have identified two main limitations of the HOTS algorithm. First, the unsupervised clustering of the kernels is highly dependent on initialization, and this can affect the performance

of the network. We have recently proposed to include a biologically plausible homeostatic gain control mechanism (Grimaldi et al., 2021). We showed there that unsupervised feature learning is qualitatively improved by balancing the activity of the different neurons within the same layer. We also tested the classification accuracy on different datasets by injecting different amounts of spatial and temporal noise into the stream of input events, demonstrating that efficiency was increased by homeostasis. The second limitation that we identified in HOTS is the classifier. Indeed, it is based on computing a histogram of the neuronal activations in the last layer of the network to perform the classification. Such a method ignores the fine-grained temporal dynamics of the stream of events produced by the final layer of the network. More importantly, a major drawback is that the classification can only be performed *post hoc*, once all the events of the tested sample have been received.

This offline layer was chosen because it provides an accurate response once all events have been emitted. In this study, to achieve a full end-to-end event-driven and online pattern categorization, we incorporate and test an online classification method in the final layer of the network. We formally demonstrate that the overall structure of the proposed model corresponds to a biologically plausible spiking neural network (SNN). To our knowledge, it is the first always-on, event-driven object recognition method, and we validate it on different datasets designed for categorizing symbol (Serrano-Gotarredona and Linares-Barranco, 2015), digit (Orchard et al., 2015a) or gesture (Amir et al., 2017) categorization. Given the simplicity of the proposed network's architecture, its event-based formalism and its local learning rules, this method is easily transferable to neuromorphic hardware to exploit the efficiency of event-based computing.

From this perspective, the structure of this paper will be 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 adding a simple biologically plausible online classification layer. We prove that our method corresponds to a SNN with Leaky Integrate-and-Fire (LIF) neuron models, which can be implemented in a neuromorphic chip. Finally, we show the quantitative improvements of the resulting classification performance, and how its dynamics can vary for different datasets. We have tested the model on different event camera datasets, and a full implementation of this algorithm is available at `https://github.com/AntoineGrimaldi/hotsline`. These scripts allow all results presented in this paper to be reproduced, and we provide links to reproducible notebooks within the text.

2. Materials and methods

In this section, we first describe the datasets used in this study and present the method we designed to test the robustness of our algorithm to spatial and temporal jitter. After giving an overview of existing object recognition algorithms, we generalize the event-based HOTS model, already described in Lagorce et al. (2017), and extend its formalism to the continuous time domain. We then present the improvements that make the algorithm fully online and biologically plausible. We introduce the homeostasis regulation rule, which allows for a better learning of the weights of the different layers (Grimaldi et al., 2021), and we describe a new classifier using Multinomial Logistic Regression (MLR) to propose an end-to-end event-driven classification algorithm. We conclude 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 (Lenz et al., 2021). It currently provides the ability to load 12 different event-based vision datasets and is based on the PyTorch language (Paszke et al., 2019). This allows to load event streams in a standard way 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 where 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 for OFF and ON events respectively. Of these datasets, 6 are labelled for object classification tasks. We choose to test the performance of our method on 3 different datasets:

- **Poker-DVS dataset** (Serrano-Gotarredona and Linares-Barranco, 2015), one of the first publicly available DVS recordings from a real-world scene that was used to test the performance of HOTS Lagorce et al. (2017). It consists of 131 occurrences of the four different symbols of playing poker cards (clubs, diamonds, hearts and spades).
- **N-MNIST dataset** (Orchard et al., 2015a), a widely used dataset that was recorded by moving an event-based camera in front of a screen onto which digitized MNIST digits (LeCun et al., 1998) were projected.
- **DVS128 Gesture dataset** (Amir et al., 2017), which consists of more complex and naturalistic recordings of real-world scenes. In this dataset, 29 subjects perform hand and arm 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. The samples provided by the dataset are 6 seconds long and, to reduce the computational load of the training, we keep only the first 3 seconds of the 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 spatial or temporal jitter to be added to the input stream. Since the relevant information is supposed to be represented within the timing and position of the input events, we can assume that the classification performance should deteriorate 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 consists of 1000 randomly selected digits with a balanced number of samples between each class.

2.2. Related work

Poker-DVS is included as it was tested using the original HOTS method (Lagorce et al., 2017). Due to its small size, this experiment acts as a toy model to test the different methods. In this paper, we focus on performing pattern recognition. A widely used event-based dataset: N-MNIST (Orchard et al., 2015a). Some related approaches have used standard artificial neural networks that have been converted to SNN, resulting in overall good classification results (Neil and Liu, 2016; Patino-Saucedo et al., 2020). In 2020, Patino-Saucedo et al. (2020) 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 to SNN (Shrestha and Orchard, 2018; Lee et al., 2016; Wu et al., 2018). More recently, it has been proposed to introduce biomimetic saccades to improve object recognition (Yousefzadeh et al., 2018). 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 has been developed in previous studies (Thiele et al., 2018; Giannone et al., 2020; Zhou et al., 2021). However, the model proposed by Zhou et al. (2021) accumulates spikes as input to reconstruct an image frame, Giannone et al. (2020) uses a sample-and-hold approach, and freezes events during a defined time step. Thiele et al. (2018) proposes to use Spike-Timing Dependent Plasticity (STDP) for unsupervised learning of spatio-temporal features. For this latter study, they also construct a supervised classifier that can learn in an online fashion and which should be able to make an inference for each event. However, they perform a classification based on the strongest response of a neuron during the time window in which the sample is presented. In this way, they do not take advantage of the event-driven nature of the signal for classification.

Then, the DVS128 Gesture dataset provides a more complex recognition task as it consists of real-world scenes with natural movements performed by subjects. Zhang et al. (2021) presents a non-local synaptic modification method with spiking and artificial neurons inspired by natural networks called self-BP. A spiking version of the deep ResNet architecture (STS-ResNet) also achieves good performance for gesture recognition as well (Samadzadeh et al., 2020). A recent study develops a bio-plausible method using SNN and STDP that achieves very good results (Safa et al., 2021). The classifier is a Support Vector Machine (SVM) applied to the feature vectors as output from the SNN. A promising method involving Spiking Recurrent Neural Networks (SRNN) is introduced in Yin et al. (2021) and achieves high online performances on several datasets. Accuracy on the DVS128 Gesture dataset reaches 97.61%, but this method requires an additional preprocessing with a convolution layer on frames computed from the DVS recording and it was not mentioned whether this number reflects an average computed on the online accuracy or not. SLAYER (Shrestha and Orchard, 2018) 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 discriminate the true class and needs to wait until the end of the output spike train to infer a decision.

For both datasets, these different methods achieve very good performances, but none of them report the accuracy as a function of the number of events used or as a function of time - thus making it impossible to derive online accuracy results. An exception in the literature is Sironi et al. (2018), which reports the accuracy as a function of the latency. It is calculated on the N-CARS dataset created for this study. However, this method uses an accumulation of time surfaces and can not perform event-by-event classification. The method we propose is the first to develop an *always-on* decision process, and we report its performance as a function of the number of events integrated by the network. We stick to the study of the three widely used datasets mentioned above (N-MNIST, DVSGesture and Poker-DVS), which can be loaded with the *tonic* package.

2.3. Event-based formalism: HOTS model

The HOTS model has three main aspects. First, it defines the core mechanism of a layer of neurons that transforms each incoming event from the event stream into a novel event (see figure 3). This layer consists of perceptron-like neurons that measure the similarity of the input to patterns stored in the neurons' synaptic weights. Crucially, this new event is selected on the basis of previous history thanks to the definition of what we will call *time surfaces* and is used as input to the current layer of the network. Secondly, the neuron that is deduced to be 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 a layer of the network. Using it as a building block, such layers can be stacked together, with the output address space of each layer defining a new input address space for the next layer. This eventually 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. This is done by reproducing the core mechanism in each layer at each position of the pixel grid. This defines weights as *kernels*, similar to Convolutional Neural Networks (CNNs). Let's 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

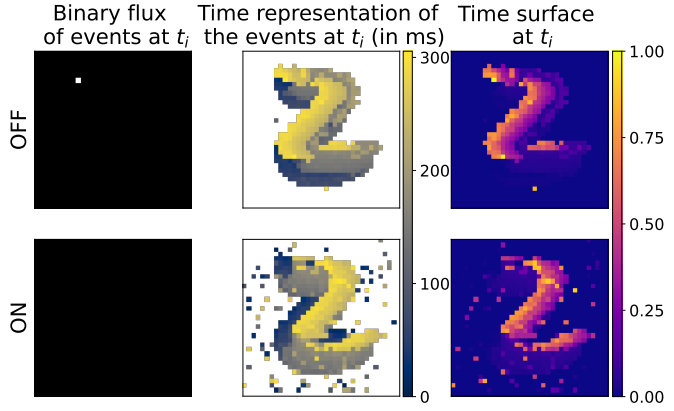

Figure 2: Illustration of the different event-based data types used in the HOTS network at a given event time. The two rows correspond to the OFF and ON polarities of the events as output of the event-based camera. (Left) Screenshot of one single event (in white) at t_i. (Middle) Timings since the latest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

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. 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 can now define 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 a so-called time context $T_a(t)$. It records the time of the last event that occurred at the specific address a before or at t, with $-\infty$ if no event was recorded (see figure 2, middle column):

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

The time context $T_a(t)$ is computed for each address at each time, forming a vector that we write $T(t)$ over the address space.

Finally, from the time context calculated at each address, we 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 which 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 assume that the layer is composed of N_n neurons which form a novel address space $\mathcal{A}$ that we can 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 with the input of this layer. These can be combined into a weight matrix $W = [w_{a,\mathbf{n}}]_{a\in\mathcal{D},\mathbf{n}\in\mathcal{A}}$. These weights are used to compute the similarity of the weight patterns with each time surface (Perrinet, 2004). 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 whose 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 mentioned 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 define 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 event stream will cascade from the first to the last layer. Each layer is defined by a weight matrix $W^\mathbf{L}$, so that each time surface will be associated with a similarity measure that generates 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 can compute a time surface for each incoming event at each layer. For this computation, we will use a different time constant $\tau^\mathbf{L}$ which 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

Now let us define the topology of the address spaces. We have seen 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, similar to the linear operation which occurs in the dendritic tree of perceptron neurons. However, from our knowledge of the early visual cortical areas, we know that the receptive field of neurons does not cover the whole visual space, but develops over a limited visual space and with stereotyped shapes. This is used in CNNs to define different *kernels* that capture the local context in the neighborhoods around each neuron. From the spatial invariance of the physical

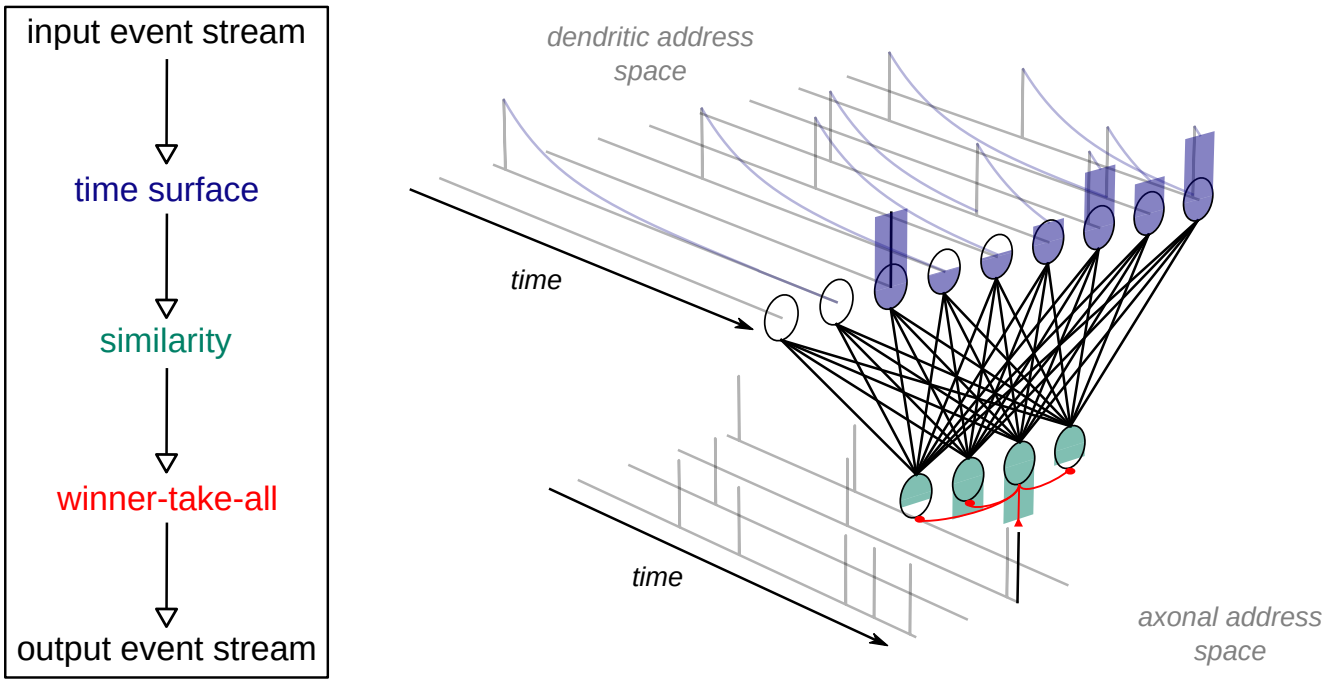

Figure 3: Illustration of the core computation made within one layer of the HOTS algorithm. On the top of the plot, we show the dendritic stream of events convolved by an exponential decay which forms the time surface. Time surfaces are computed at the timestamp of each event/spike. The time surface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

problem, it is assumed that the kernels should be similar across different positions and define a convolution operator. A notable advantage of this representation is its invariance to translations. Thus, in analogy to what is done in CNNs, we can 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 call them dendritic channels. Each address can be decomposed into its position and its dendritic channel, i.e. $a^{\mathbf{L}} = (x^{\mathbf{L}}, y^{\mathbf{L}}, \mathbf{p}^{\mathbf{L}})$. 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 can 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 convolution operator and $\sim$ is the symmetry operator, which allows the correlation in equation (3) to be computed using convolution. Note that $\tilde{K}^{\mathbf{L}}_{\mathbf{k}} * S^{\mathbf{L}}(t)$ represents the activity map and can be computed efficiently by a convolution operation. In our formalism, time surfaces are defined globally, and each weight vector corresponds to a column of the weight matrix, constructed with a Toeplitz operation, where indices are associated with 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 this radius are zero, and thus the similarity measured with the global time surface $S(t)$ will give the same results as with the locally defined time surfaces in the original HOTS formalization.

Furthermore, the HOTS algorithm, specified in Lagorce et al. (2017), 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 to be performed only on 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 over the different axonal channels, and modifying 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 result, the next layer will send 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. In summary, each layer takes input events from its previous layer and feeds events to the next layer by repeating these steps. It follows that neurons within a layer $\mathbf{L}$ compete for features: each incoming event produces a single event on the axonal space. Following what is observed in the 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 $R^{\mathbf{L}}$ so that they increase as we move up the hierarchy. 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 result, the network learns increasingly complex spatio-temporal features in a hierarchical fashion. 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 so that it is adapted to the average interspike interval on each layer: $\tau^{\mathbf{L}} = N_p^{\mathbf{L}}.\tau^{\mathbf{L}=0}$. A description of the hyperparameters associated with the experiments on the different datasets is given in 3.1.

As for learning of the weights, this is done in an unsupervised manner. During the unsupervised clustering phase, the kernels are updated with the same learning rule as described in Lagorce et al. (2017):

$$\tilde{K}^{\mathbf{L}}_{\mathbf{p}_i^{\mathbf{L}+1}} \leftarrow \tilde{K}^{\mathbf{L}}_{\mathbf{p}_i^{\mathbf{L}+1}} + \eta_{\mathbf{p}_i^{\mathbf{L}+1}} \cdot \beta_{\mathbf{p}_i^{\mathbf{L}+1}} \cdot (S^{\mathbf{L}}_{local}(t_i) - \tilde{K}^{\mathbf{L}}_{\mathbf{p}_i^{\mathbf{L}+1}})$$
$$\text{with } \eta_{\mathbf{p}_i^{\mathbf{L}+1}} = \frac{0.01}{1 + \frac{\#_{\mathbf{p}_i^{\mathbf{L}+1}}}{20000}}$$

where we define $\#_{\mathbf{p}_i^{\mathbf{L}+1}}$ as the number of times kernel $\tilde{K}^{\mathbf{L}}_{\mathbf{p}_i^{\mathbf{L}+1}}$ has been 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 bring the selected kernel $\tilde{K}^{\mathbf{L}}_{\mathbf{p}_i^{\mathbf{L}+1}}$ closer to the observed time surface. In fact, $\beta_{\mathbf{p}_i^{\mathbf{L}+1}}$ represents the product of the activation for the presynaptic and postsynaptic neurons. Note that the training of the kernels is shared among all the spatial locations for the same axonal channel, just as in CNNs. This mechanism is similar in principle to that used by the *k*-means algorithm and is implemented in many other unsupervised learning schemes (Perrinet et al., 2003). For the layers of the HOTS model, we filter the time surfaces with a threshold on the number of active pixels to avoid noisy or isolated events. We set this threshold to $2 \cdot R^{\mathbf{L}}$. Figure 4 provides 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 (Grimaldi et al., 2021). Similar regulation methods on an event-based dataset are used in Diehl and Cook (2015); Wu et al. (2019) to balance the firing rate over the neurons of each layer of the SNN. The model of Diehl and Cook (2015) uses an adaptive membrane threshold, while Wu et al. (2019) adds an auxiliary neuron per layer to regulate the firing rates. In this last paper, they make a comparison of this technique with zero-mean batch normalization (Ioffe and Szegedy), which is used for training deep neural networks. These methods are similar in their aims and are well justified in terms of efficient coding (Perrinet, 2010).

Here, we implement homeostasis regulation by adapting the heuristics used in a sparse coding scheme (Perrinet, 2019). It simply consists of modifying the similarity measure (see equation (3)) as follows:

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

Where we use the same gain as defined in Grimaldi et al. (2021):

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

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 this layer. Note that the gain control is applied to each map of kernel activities, not to the activity of individual neurons, due to the translation invariance property of the architecture. This control rule allows the different kernels to be trained in such a way that the response of only a few of them is avoided, reaching an equilibrium when $f_{\mathbf{k}}(t) = \frac{1}{N}$, i.e. when they are on average equally likely to be activated.

In practice, we observed that adding homeostasis leads to a better clustering of the weight matrices, see figure 4. Note that during the unsupervised clustering phase, the neural activity is balanced across all digits. The homeostasis process does not necessarily result in an equi-probable neural activity for one digit, but over the whole learning set, in line with the efficient coding hypothesis (Barlow et al., 1961). It also avoids introducing an *ad hoc* heuristics into the learning rule to achieve convergence for all neurons. For example, in Lagorce et al. (2017), 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 at a time. In this work, the weights are initialized randomly, and we allow spikes to feed each layer of the network even if a given layer is not fully trained, but convergence is still robust. As a result, this additional ingredient in the unsupervised learning phase makes our algorithm behave more like living systems.

2.5. Online event-based classification

In the original HOTS algorithm, classification is performed by comparing the activation histograms across the channels of the last layer of the network with the average observed for each given class. This classification by histogram comparison is performed *post hoc*, after the encoding of an element from the dataset. Here, we introduce a novel online classification scheme, that is, where classification is performed for each spike that reaches the classifier, and more generally at any time when a classification is required. Following the same strategy used for the construction of time surfaces, each event reaching the last layer $\mathbf{L} = \mathbf{C}$ of the network can indeed be transformed into a time surface $S^{\mathbf{C}}(t)$ using a time constant $\tau_{\mathbf{C}}$. This constant can vary from one dataset to another according to the statistics of the samples. The time surface thus forms an analog vector that can be used in a Multinomial Logistic Regression (MLR) model to achieve supervised classification. Such MLR models are used, for example, in the last layer of classical deep learning networks (Lecun et al., 1998) and are compatible with a neural implementation (Berens et al., 2012). More specifically, it corresponds to the similarity measure (see equation (3)) of the MLR weights with the input, stacked with a sigmoid nonlinearity. The weights are defined over the whole dendritic space, i.e. there is no local context as it was defined for the

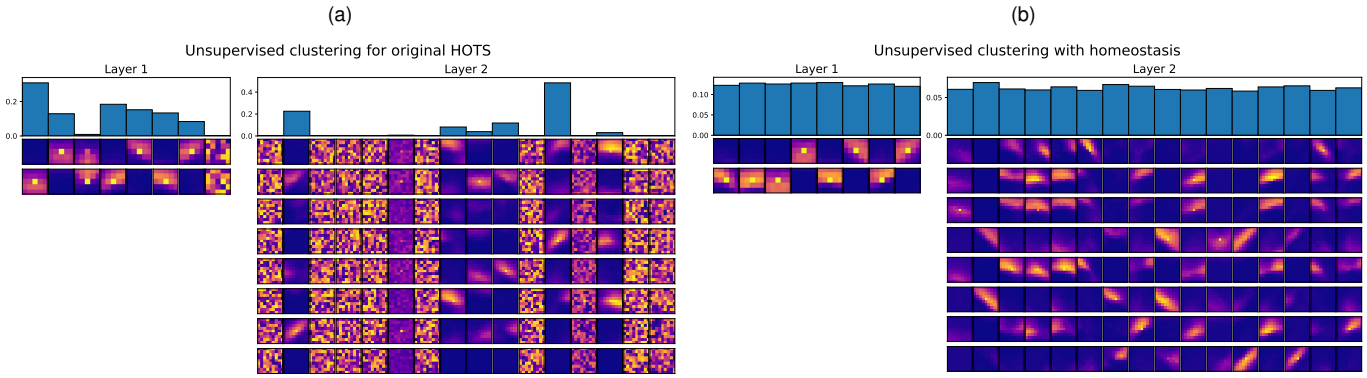

Figure 4: Activation histograms and time surfaces obtained in the unsupervised learning algorithm (a) for the original HOTS network (replicated from Lagorce et al. (2017) with time surfaces intialized randomly) and (b) for the bio-plausible version with homeostasis. Activation histograms correspond to 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.

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 with 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:

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

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 MLR model weights associated with the class c. The final prediction can be made for each incoming event using 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 through the maximum value of the similarity measure. We obtain the same spiking process as in any layer of the HOTS network:

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

The result is an always-on decision process, that can make a prediction at any time. In the following, we will perform event-driven prediction and compare the classification results 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 probabilities are calculated for each event, the class prediction can only be made for some events with a probability above a defined threshold. This flexibility to make predictions with a specific confidence threshold allows performance to be improved while maintaining an event-driven approach to computation.

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 each time surface along with its true class as supervision pairs. The MLR model was implemented using the PyTorch language, and training was performed using a gradient descent with the Adam optimizer. Our loss function is the binary cross entropy computed on the output spike train, and the learning parameters are described in 3.1. Once the MLR model was trained, we obtained analog vectors from the hierarchical network computations on the test set. We then tested classification performances by sending these vectors to the MLR model, which outputs the probability of each class being true. The decision process can be the arg max function of the probability values, and this 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 show that, when it is extended to the continuous-time domain, this algorithm can be implemented as a SNN. Indeed, the definition of the time surface modulated by an exponential decay in equation (2) bears an analogy to the LIF model with exponentially decaying postsynaptic potentials, as described for other SNNs (Rueckauer et al., 2017). We aim to describe the event-based model on a time continuum thanks to Ordinary Differential Equations (ODE) and to bridge such an algorithm with the SNN framework from computational neuroscience.

2.6.1. HOTS as a SNN

Let's look at 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 each time using equation (2). Looking at figure 3, one can see 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 (Gerstner, 1995) 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 on the right hand side of equation (8) is a modulated Dirac function that implements the integration of a new presynaptic potential at $t = t_i$. The modulation $1 - S_a(t)$ is such that at the moment of the event, the new value of the potential becomes $S_a(t) + (1 - S_a(t)) = 1$. This 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 effect on activity, as implemented in the definition of the time context. As a consequence, it implements a kind of reset mechanism that allows the time surface to be computed 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 corresponding to the integration of synaptic inputs in 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 obtain 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 can be simplified to a sum of 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 an ODE is classical for describing the evolution of the membrane potential of LIF neurons. Note that the main change is the modulation of the integration of incoming spikes, which allows only the time to the last spike to be represented. The hierarchical network proposed in Lagorce et al. (2017) is then equivalent to a SNN composed of LIF neurons with a Hebbian-like learning mechanism, as mentioned in section 2.3.3. In this SNN, for each incoming event from the event-based camera, one spike is emitted for each layer of the network. This 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 to a SNN implementation has already been drawn in Berens et al. (2012). Analogous to biology and as described in section 2.6.1, the linear combination of the input time surface with the MLR weights corresponds to the integration of presynaptic spikes on the dendritic tree of a postsynaptic neuron associated with a class. Then, $\beta_c(t) = \langle W_c^{\mathbf{C}}, S(t) \rangle$ represents the membrane potential of the postsynaptic neuron associated with 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. (2013) showed 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 complete inhibition of other neurons until the next decision. As a consequence, if the classification is event-driven, only one spike will be emitted for the most probable class only for each event. The spiking mechanism of the classification layer is then 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 an event of rank i 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 with 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 paper can be fully described by a SNN. The learning of the weights is done in an event-driven manner and corresponds to Hebbian-like mechanisms for the neurons, both inside the network and in the classification layer. We claim that these local learning rules are advantageous both in terms of bio-plausibility and for energy-efficient on-chip implementations (Roy et al., 2019).

3. Results

The classification results obtained with our method are presented in this section. First, we present the online classification performance of the network, which is the main novelty of our study. We then compare the performance with 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 robustness of our algorithm to both temporal and spatial jitter, comparing it to the original

method proposed in Lagorce et al. (2017). Let's start with 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. The parameter tuning for $\mathbf{L}_1$ and $\mathbf{L}_2$, which are layers similar to those in the original HOTS network, was done on subsets of the different datasets by computing the accuracy for each of the different architectures using histogram comparison as done in Lagorce et al. (2017). For each dataset, the classification layer is implemented in PyTorch, and we train it using gradient descent and the Adam optimizer. The time surfaces as input to 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 performance of the classifier with different parameters, on a subset of the original dataset. For each dataset, we set the number of epochs to 33 and keep the optimizer's default parameters. For both the N-MNIST and DVSGesture datasets, to reduce the number of computations and to avoid reaching a local minimum within the first samples, we perform the learning only on a randomly chosen percentage of the computed time surfaces. We keep 10% and 5% of the time surfaces of a sample for N-MNIST and DVSGesture datasets, respectively. For DVSGesture, we also apply spatial downsampling by a factor of $2 \cdot R^{\mathbf{L}} + 1$ for each dimension and at each layer. With the winner-takes-all spiking mechanism used in the unsupervised layers and the spatial downsampling, the core layers of HOTS now implement an event-based convolution with a max-pooling. This allows reducing the dimensions of the feature maps and the amount of computations performed in these recordings with a greater number of events and a wider pixel grid.

	$\mathbf{L}_1$	$\mathbf{L}_2$	**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	$N_K = 16$ $R = 2$ $\tau = 10$ ms	$N_K = 32$ $R = 2$ $\tau = 160$ ms	$\eta = 0.0001$ $\tau_{\mathbf{C}} = 1$ s $\theta = 0.4$

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

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

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 accuracy value 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 condition 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 to make a decision. In this last condition, in order to filter out events in periods of poor information content (especially at the beginning), we select only events with a decision confidence above a threshold. This improves the average performance of the classification. However, it introduces some time delay to accumulate enough evidence to make a prediction. We also report the accuracy values for the classification *post hoc* with a k-nearest neighbors algorithm on the activation histograms. Two results are shown in figure 5, one for the *original HOTS* 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 vary. We set a maximum number of events to represent the accuracy by taking the 90th percentile of the dataset in terms of number of events. The accuracy of the event-based classification for each dataset is shown in Figure 5. Note that the x-axis is plotted on a logarithmic scale and that only a small number of events will allow for significant classification above chance.

In Figure 5-(a), we observe the online inference 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 total 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. If we set a confidence threshold on the MLR, we obtain a perfect classification once at least 35 events are received. Given the small number of samples in this dataset, we evaluate the performance of the network on two more complex and widely used datasets.

The online accuracy on the N-MNIST dataset is shown 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 the performance for the clustering phase, and we recall that the main advantage of this regularization is to reduce the sensitivity of the unsupervised learning to the initialization (Grimaldi et al., 2021). For *online HOTS*, we observe an accuracy above chance after the very first events, which increases with the number of events received by the network. Note that the accuracy value increases drastically after about 1000 events and reaches values

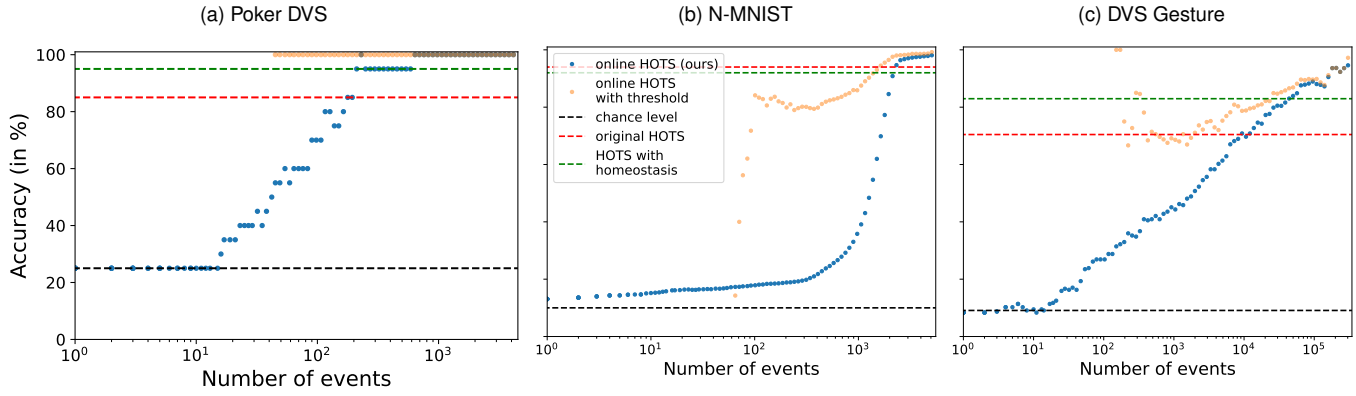

Figure 5: Accuracy for online classification on 3 different datasets (see text for details).

above the original method approximately at 2000 events, the average 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 get an accuracy of 70.1% and 96.6% for the accuracy calculated when the decision is made at the timestamp of the last event. Another way of calculating the *post hoc* accuracy with this probabilistic approach is to choose the decision that was made with the highest confidence. This gives us an accuracy of 97.4%, which is close to the SOTA (see next 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 only be obtained after a minimum of about 100 events, in line with the idea of ultra-fast categorization. With this last decision method, we obtain an average accuracy of 96.2% which is greatly improved compared to the mean performance over all the decisions without confidence threshold. The results of 5-(b) indicate that the second and the third saccades 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 a small improvement when the other saccades are also used (Lee et al., 2016; Frenkel et al., 2020; Thiele et al., 2018).

For the DVSGesture dataset, we confirm the improvement of our method over 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 about 100 events. More evidence needs to be accumulated, and then the accuracy increases monotonically, outperforming the previous method after about 10.000 events (an average of 9.3% of the sample). These event-based recordings have a much higher event density than the other datasets, and we remind the reader that only 3 seconds of the recording is kept to test our algorithm. The average accuracy for all the decisions is 85.7% and 87.2% when the decision is taken at the last event received. The average always-on accuracy can reach 88.8% by setting the confidence threshold to 0.4. When we make a decision *post hoc*, choosing the classifier output with the highest probability, we get 89.8%.

3.2. Comparison to the state-of-the-art

To compare the performance of our method with the SOTA, we choose to compute the accuracy when the decision is made with the highest confidence, as 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, preprints are discarded, and we focus on event-based methods that achieve the best performance. We split the table 2 into two different parts for the methods that

	N-MNIST	DVS Gesture
HOTS (with k-NN) (Lagorce et al., 2017)	94.39%	83.0%
HATS (Sironi et al., 2018)	99.1%	–
SLAYER (Shrestha and Orchard, 2018)	99.2%	93.64%
Spike-based BP (Fang et al., 2021)	99.61%	97.57%
DSNN-STDP (Thiele et al., 2018)	95.77%	–
DECOLLE (Kaiser et al., 2020)	96%	95.54%
self-BP (Zhang et al., 2021)	–	84.76%
hybrid CNN-SRNN (Yin et al., 2021)	–	97.61%
Ours	**97.4%**	**89.8%**

Table 2: Offline classification accuracy for the N-MNIST and DVSGesture datasets. The upper part of the table corresponds to non-biologically plausible algorithms, and the lower part to biologically plausible ones.

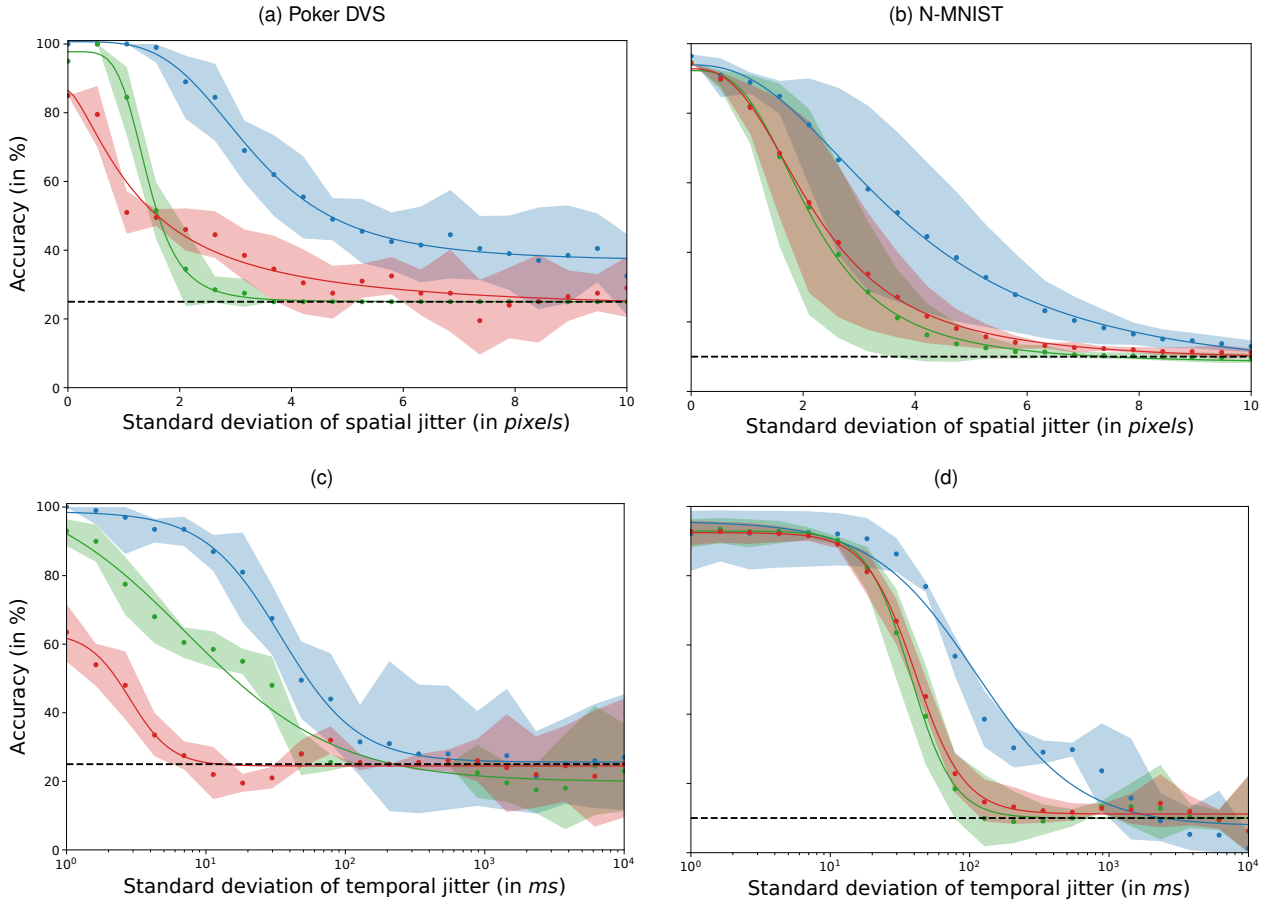

Figure 6: Evolution of classification accuracy as a function of (a-b) spatial and (c-d) temporal jitter.

are biologically plausible (bottom) and the others (top). We skip the comparison of the results obtained with the Poker DVS dataset, which serves 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-layer feedforward network structure with a bio-plausible learning achieves 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 of this event-driven object recognition method. To do this, we perturb the original datasets by adding temporal or spatial jitter to the events. Jitter is applied only to the test 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 to the test samples. For each amount of jitter applied to the test set, 10 repetitions are performed to obtain different accuracy values. Finally, we fit a beta distribution to each of these results to compute the percentiles shown in Figure 6. To compare with previous results obtained in Grimaldi et al. (2021), we plot the offline accuracy obtained when making one decision per sample. We reduce the number of computations for this analysis by using subsets of the N-MNIST dataset (1000 samples). For each amount of jitter applied to the test subset, 10 repetitions are performed to obtain different accuracy values. Finally, we fit a beta distribution to these results to compute the percentiles shown in Figure 6. As the proposed method is a proof of concept for event-based computation, the simulations on GPU are not optimized and results with jitter applied to the DVSGesture dataset are not computed for reasons of simulation time. We highlight the fact, that to our knowledge, no other studies have performed this test on event-based recordings. Simulated on two different DVS datasets, these results provide insight into the robustness of our algorithm, but also highlight the features that are relevant for classification within the event-based recordings.

As expected, the higher the jitter, the greater the negative impact on classification. The decrease in accuracy as a function of jitter fits well to a sigmoid function that decreases from a maximum accuracy value to reach the chance level. Using this fit, it is possible to define a critical standard deviation of jitter in pixels or in ms where the accuracy drops to half of its maximum compared to the chance level. This half saturation level provides a signature value for the relevant information contained in the signal.

Figure 6-(a) shows the evolution of the accuracy for the different methods as a function of the amount of spatial jitter applied to the PokerDVS dataset. Accuracy reaches half sat-

uration for a spatial jitter with a standard deviation of 1.50, 1.66 and 3.87 pixels, for *HOTS*, *HOTS with homeostasis* and *online HOTS* respectively. Figure 6-(b) shows results for the N-MNIST dataset. Curves for *HOTS*, *HOTS with homeostasis* are very close but show slightly different half saturation levels: 2.42 pixels for *HOTS*, 2.32 and for *HOTS with homeostasis*. The homeostasis itself does not give significant improvement in this particular example, in line with the results obtained in Figure 5-(b). However, *online HOTS* shows overall significantly better performances and reaches its half saturation level at 3.74 pixels Reaching half saturation level at approximately a standard deviation of the spatial jitter equal to 3.8 pixels demonstrates that this method relies heavily on the spatial information. However, considering the small pixel grid of these datasets (32×32 for PokerDVS and 28×28 for N-MNIST) and the gradual decrease of the accuracy, we observe robustness to spatial jitter for the different methods. In any case, the results demonstrate a significant improvement of the robustness to spatial jitter for the new 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. The 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 one standard deviation of the jitter distribution in ms. For *HOTS*: 1.77 ms; *HOTS with homeostasis*: 8.16 ms; and *online HOTS*: 34.5 ms. For N-MNIST (see figure 6-(d)), the half saturation values are 49.77 ms, 41.32 ms and 126.2 ms for *HOTS*, for *HOTS with homeostasis* and for the algorithm presented in this study respectively. Even the original method offers a high resilience to temporal jitter compared to the duration of the recordings. For PokerDVS, this resilience increases significantly with the addition of the homeostatic gain control but not for N-MNIST where the robustness curves are, again, very similar. For the *online HOTS* method, we observe an increased robustness to temporal jitter. The standard deviation of the added jitter must reach a similar timescale as the recording itself (5 times the average duration of a recording for PokerDVS and one third of the average duration for N-MNIST samples). This surprisingly high robustness may be due to the use of time surfaces to encode the signal. When temporal jitter is added, the locations of events are preserved and only the timing is affected. A time surface with the same spatial structure is computed from a jittered or non-noisy signal. By applying an exponential decay to the delays, the effect of jitter is reduced. This time surface is then compared to smooth time surfaces with a scalar product over the entire spatial window. This technique makes the encoding of input events more robust to local temporal variations. The increase in robustness for the homeostatic gain methods may be due to the improved clustering of the network's time surfaces. The way we construct the analog vector as input to 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 a few events at 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 when the time surface is computed. The higher the time constant, the better the resolution for the recent past history, but the more events can accumulate on the same 2D time surface, interfering with accurate 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 similar to a SNN. 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 (Lennie, 1998). Furthermore, cortical areas were found to follow a hierarchical order of intrinsic time scales (Murray et al., 2014). 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 computations performed by higher level areas. This particular organization of the HOTS architecture and the evolution of the temporal surface parameters 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 generic and robust. Second, we added an online classification layer that performs MLR and is compatible with a neural implementation (Berens et al., 2012). As shown in section 2.6, the learning rules are local and Hebbian-like. This makes the learning of the network easily transferable to neuromorphic hardware. Once trained, the network can perform an always-on classification, i.e. it can infer a prediction whenever necessary. 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 number 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 that require fast or accurate decisions. Overall, these results provide a good illustration of the potential synergy between neuromorphic engineering and computational neuroscience.

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”).

During the preparation of this work the authors used DeepL Write in order to improve language and readability. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, C., Nayak, T., Andreopoulos, A., Garreau, G., Mendoza, M., Kusnitz, J., Debole, M., Esser, S., Delbruck, T., Flickner, M., Modha, D., 2017. A low power, fully event-based gesture recognition system, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Bardow, P., Davison, A.J., Leutenegger, S., 2016. Simultaneous optical flow and intensity estimation from an event camera, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 884–892.

Barlow, H.B., et al., 1961. Possible principles underlying the transformation of sensory messages. Sensory communication 1.

Benosman, R., Clercq, C., Lagorce, X., Ieng, S.H., Bartolozzi, C., 2013. Event-based visual flow. IEEE transactions on neural networks and learning systems 25, 407–417.

Berens, P., Ecker, A.S., Cotton, R.J., Ma, W.J., Bethge, M., Tolias, A.S., 2012. A fast and simple population code for orientation in primate V1. J Neur 32. URL: https://www.jneurosci.org/content/32/31/10618, doi:10.1523/JNEUROSCI.1335-12.2012.

Boahen, K.A., 2000. Point-to-point connectivity between neuromorphic chips using address events. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 47, 416–434.

Diehl, P.U., Cook, M., 2015. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. Frontiers in computational neuroscience 9, 99.

Fang, W., Yu, Z., Chen, Y., Masquelier, T., Huang, T., Tian, Y., 2021. Incorporating learnable membrane time constant to enhance learning of spiking neural networks, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 2661–2671.

Frenkel, C., Legat, J.D., Bol, D., 2020. A 28-nm convolutional neuromorphic processor enabling online learning with spike-based retinas, in: 2020 IEEE International Symposium on Circuits and Systems (ISCAS), IEEE. pp. 1–5.

Gallego, G., Delbruck, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A., Conradt, J., Daniilidis, K., et al., 2019. Event-based vision: A survey. arXiv preprint arXiv:1904.08405 .

Gallego, G., Lund, J.E., Mueggler, E., Rebecq, H., Delbruck, T., Scaramuzza, D., 2017. Event-based, 6-dof camera tracking from photometric depth maps. IEEE transactions on pattern analysis and machine intelligence 40, 2402–2412.

Gerstner, W., 1995. Time structure of the activity in neural network models 51, 738–758. URL: https://link.aps.org/doi/10.1103/PhysRevE.51.738, doi:10.1103/PhysRevE.51.738.

Giannone, G., Anoosheh, A., Quaglino, A., D'Oro, P., Gallieri, M., Masci, J., 2020. Real-time classification from short event-camera streams using input-filtering neural odes. arXiv preprint arXiv:2004.03156 .

Grimaldi, A., Boutin, V., Ieng, S.H., Perrinet, L.U., Benosman, R., 2021. A homeostatic gain control mechanism to improve event-driven object recognition, in: Content-Based Multimedia Indexing (CBMI) 2021. URL: https://laurentperrinet.github.io/publication/grimaldi-21-cbmi/.

Hidalgo-Carrió, J., Gehrig, D., Scaramuzza, D., 2020. Learning monocular dense depth from events. arXiv preprint arXiv:2010.08350 .

Ioffe, S., Szegedy, C., . Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. URL: http://arxiv.org/abs/1502.03167, arXiv:1502.03167.

Kaiser, J., Mostafa, H., Neftci, E., 2020. Synaptic plasticity dynamics for deep continuous local learning (decolle). Frontiers in Neuroscience 14, 424.

Kim, H., Leutenegger, S., Davison, A.J., 2016. Real-time 3d reconstruction and 6-dof tracking with an event camera, in: European Conference on Computer Vision, Springer. pp. 349–364.

Lagorce, X., Orchard, G., Galluppi, F., Shi, B.E., Benosman, R.B., 2017. HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 39, 1346–1359. URL: http://www.ncbi.nlm.nih.gov/pubmed/27411216http://ieeexplore.ieee.org/document/7508476/, doi:10.1109/TPAMI.2016.2574707.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324.

Lecun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324. doi:10/d89c25.

Lee, J.H., Delbruck, T., Pfeiffer, M., 2016. Training deep spiking neural networks using backpropagation. Frontiers in neuroscience 10, 508.

Lennie, P., 1998. Single units and visual cortical organization. Perception 27, 889–935.

Lenz, G., Chaney, K., Shrestha, S.B., Oubari, O., Picaud, S., Zarrella, G., 2021. Tonic: event-based datasets and transformations. URL: https://doi.org/10.5281/zenodo.5079802, doi:10.5281/zenodo.5079802. Documentation available under https://tonic.readthedocs.io.

Murray, J.D., Bernacchia, A., Freedman, D.J., Romo, R., Wallis, J.D., Cai, X., Padoa-Schioppa, C., Pasternak, T., Seo, H., Lee, D., et al., 2014. A hierarchy of intrinsic timescales across primate cortex. Nature neuroscience 17, 1661–1663.

Neil, D., Liu, S.C., 2016. Effective sensor fusion with event-based sensors and deep network architectures, in: 2016 IEEE International Symposium on Circuits and Systems (ISCAS), IEEE. pp. 2282–2285.

Nessler, B., Pfeiffer, M., Buesing, L., Maass, W., 2013. Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. PLoS Comput Biol 9, e1003037.

Orchard, G., Jayawant, A., Cohen, G.K., Thakor, N., 2015a. Converting static image datasets to spiking neuromorphic datasets using saccades. Frontiers in neuroscience 9, 437.

Orchard, G., Meyer, C., Etienne-Cummings, R., Posch, C., Thakor, N., Benosman, R., 2015b. Hfirst: a temporal approach to object recognition. IEEE transactions on pattern analysis and machine intelligence 37, 2028–2040.

Osswald, M., Ieng, S.H., Benosman, R., Indiveri, G., 2017. A spiking neural network model of 3d perception for event-based neuromorphic stereo vision systems. Scientific reports 7, 1–12.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al., 2019. Pytorch: An imperative style, high-performance deep learning library. arXiv preprint arXiv:1912.01703 .

Patino-Saucedo, A., Rostro-Gonzalez, H., Serrano-Gotarredona, T., Linares-Barranco, B., 2020. Event-driven implementation of deep spiking convolutional neural networks for supervised classification using the spinnaker neuromorphic platform. Neural Networks 121, 319–328.

Pérez-Carrasco, J.A., Zhao, B., Serrano, C., Acha, B., Serrano-Gotarredona, T., Chen, S., Linares-Barranco, B., 2013. Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward convnets. IEEE transactions on pattern analysis and machine intelligence 35, 2706–2719.

Perrinet, L.U., 2004. Feature detection using spikes : the greedy approach. Journal of Physiology-Paris 98, 530–9. URL: http://dx.doi.org/10.1016/j.jphysparis.2005.09.012, doi:10.1016/j.jphysparis.2005.09.012.

Perrinet, L.U., 2010. Role of homeostasis in learning sparse representations. Neural Computation 22, 1812–36. URL: https://arxiv.org/abs/0706.3177, doi:10.1162/neco.2010.05-08-795.

Perrinet, L.U., 2019. An adaptive homeostatic algorithm for the unsupervised learning of visual features. Vision 3, 47. URL: https://spikeai.github.io/HULK/, doi:10.3390/vision3030047.

Perrinet, L.U., Samuelides, M., Thorpe, S.J., 2003. Emergence of filters from natural scenes in a sparse spike coding scheme. Neurocomputing 58–60,

821–6. URL: http://dx.doi.org/10.1016/j.neucom.2004.01.133, doi:10.1016/j.neucom.2004.01.133.

Roy, K., Jaiswal, A., Panda, P., 2019. Towards spike-based machine intelligence with neuromorphic computing. Nature 575, 607–617.

Rueckauer, B., Lungu, I.A., Hu, Y., Pfeiffer, M., Liu, S.C., 2017. Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification. Frontiers in Neuroscience 11.

Safa, A., Sahli, H., Bourdoux, A., Ocket, I., Catthoor, F., Gielen, G.G.E., 2021. Learning event-based spatio-temporal feature descriptors via local synaptic plasticity: A biologically-realistic perspective of computer vision. CoRR abs/2111.00791. URL: https://arxiv.org/abs/2111.00791, arXiv:2111.00791.

Samadzadeh, A., Far, F.S.T., Javadi, A., Nickabadi, A., Chehreghani, M.H., 2020. Convolutional spiking neural networks for spatio-temporal feature extraction. CoRR abs/2003.12346. URL: https://arxiv.org/abs/2003.12346, arXiv:2003.12346.

Serrano-Gotarredona, T., Linares-Barranco, B., 2015. Poker-dvs and mnist-dvs. their history, how they were made, and other details. Frontiers in neuroscience 9, 481.

Shrestha, S.B., Orchard, G., 2018. Slayer: Spike layer error reassignment in time. Advances in neural information processing systems 31.

Sironi, A., Brambilla, M., Bourdis, N., Lagorce, X., Benosman, R., 2018. HATS: Histograms of averaged time surfaces for robust event-based object classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1731–1740.

Thiele, J.C., Bichler, O., Dupret, A., 2018. Event-based, timescale invariant unsupervised online deep learning with stdp. Frontiers in computational neuroscience 12, 46.

Tschechne, S., Sailer, R., Neumann, H., 2014. Bio-inspired optic flow from event-based neuromorphic sensor input, in: IAPR Workshop on Artificial Neural Networks in Pattern Recognition, Springer. pp. 171–182.

Wu, Y., Deng, L., Li, G., Zhu, J., Shi, L., 2018. Spatio-temporal backpropagation for training high-performance spiking neural networks. Frontiers in neuroscience 12, 331.

Wu, Y., Deng, L., Li, G., Zhu, J., Xie, Y., Shi, L., 2019. Direct training for spiking neural networks: Faster, larger, better, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 1311–1318.

Yin, B., Corradi, F., Bohté, S.M., 2021. Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. Nature Machine Intelligence 3, 905–913.

Yousefzadeh, A., Orchard, G., Serrano-Gotarredona, T., Linares-Barranco, B., 2018. Active perception with dynamic vision sensors. Minimum saccades with optimum recognition. IEEE transactions on biomedical circuits and systems 12, 927–939.

Zhang, T., Cheng, X., Jia, S., Poo, M.m., Zeng, Y., Xu, B., 2021. Self-backpropagation of synaptic modifications elevates the efficiency of spiking and artificial neural networks. Science Advances 7, eabh0146.

Zhou, S., Wang, W., Li, X., Jin, Z., 2021. A spike learning system for event-driven object recognition. arXiv preprint arXiv:2101.08850 .

Zhu, A.Z., Chen, Y., Daniilidis, K., 2018. Realtime time synchronized event-based stereo, in: Proceedings of the European Conference on Computer Vision (ECCV), pp. 433–447.