In neuronal models, an efficient use or detection of these spatio-temporal patterns embedded in the spike train comes with the integration of heterogeneous delays [**?**, **?**, **?**]. Notably, Izhikevich [**?**] introduced the notion of the polychronous group as a repetitive motif of spikes defined by a subset of neurons with different, yet precise, relative spiking delays. This representation has a much greater information capacity in comparison to a firing-rate based neural coding approach through the variety of configurations and the possible coexistence of multiple superposed motifs. However, most current neuroscience-inspired computer vision algorithms (for instance convolution neural networks) do not make use of this dynamic aspect. A novel emerging representation is that provided by event-based cameras, in which each pixel independently processes its input and emits an event for positive or negative increments of the log-luminance. The shift from the classical, dense representations to this sparse encoding of visual information offers a better analogy to neurobiology, but also offers more energy-efficient computations. Yet, flexible and robust event-driven algorithms for classical computer vision tasks, such as motion detection, are still not able to compete with the state-of-the-art of dense computer vision solutions.

In this work, we study the emergence of such spatio-temporal spiking motifs when training a single layer of spiking neurons on a supervised classification task (see Fig. **??**). We develop a SNN-like method able to learn hetero-synaptic delays to perform motion detection on a synthetic event-based dataset. Because neuromorphic devices are, by design, good candidates to integrate computations with time, we highlight the fact that this event-driven algorithm is transferable to such hardware.

1 Methods

In a recent study, we have introduced a classification model applied to event streams based on Multinomial Logistic Regression (MLR) [**?**]. To represent temporal relationships between events, this model first builds "time surfaces", an image computed using the time difference to the last recorded events [**?**]. By transforming each event in the stream as a vectorial input, this MLR classifier is able to make a decision for every single event. We have demonstrated on several datasets that it provides with online computations resulting in ultra-fast classification. Additionally, we made a formal bridge between the event-based MLR and a SNN, demonstrating the bio-plausibility of this method and its possible integration to neuromorphic hardware.

Here, we propose to extend such a model to a layer of spiking neurons which include, in addition to synaptic weights, hetero-synaptic delays. In particular, one afferent may be connected with multiple delays and, crucially, we will explicitly use the delay as a computational process. The objective in this model, by including the dimension of temporal delays, is to increase the representational capacities of the classifier. In the perspective of building energy-efficient algorithms, we will also titrate quantitatively the best trade-off between robustness and computation time when increasing the number of these hetero-synaptic delays.

1.1 Task definition: fast motion detection

To test our model, we will quantify its ability to categorize different motions. In that order, we use a set of synthetic visual stimuli, *Motion Clouds* [**?**] which are natural-like random textures for which we can control for velocity, among other parameters (see Fig. **??**). In particular, we will set the spatial size to $(N_{\mathrm{X}}, N_{\mathrm{Y}})$ and consider a discretization of time with a time step of 1 such that $t \in \mathbb{N}$. Movies' duration are here set to $N_{\mathrm{t}} = 400$. This procedure defines a set of textures with different spatial properties and different motions $\vec{\mathbf{v}}_k$ with $1 \leq k \leq N_{\mathrm{class}}$ and $N_{\mathrm{class}} = 8$ defined by a constant speed and linearly spaced directions: $v_k = (v \cdot \cos\left(2\pi \cdot \frac{k}{N_v}\right), v \cdot \sin\left(2\pi \cdot \frac{k}{N_v}\right))$ (see Fig. **??**-(a) for an illustration). For any given velocity, we also varied the parameters of the textures, such as the mean and variance of the orientation or spatial frequency content. This method provides a rich dataset of textured movies for which we know the ground truth for motion.

To transform each movie into events, we compute a gradient image (initialized at zero) by adding the gradient of the pixels' intensity over two successive frames. If, on a specific pixel at that specific timestamp, the absolute value of this

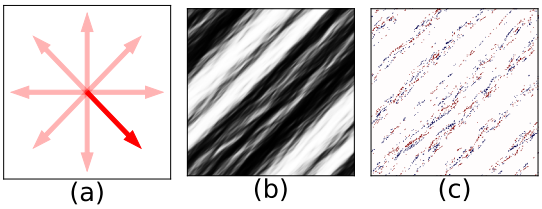

Fig. 1: **Motion detection task. (a)** The motion direction represented as the plain red vector, other possible motion directions are represented in light red. **(b)** A screenshot of one generated naturalistic textured stimulus at a specific time. **(c)** The corresponding ON (in red) and OFF (in blue) event stream generated from the stimuli on (b) and constituting the input to the spiking neural network.

gradient exceeds a threshold, an event is generated. The event has either an OFF or ON polarity, respectively whether the gradient is negative or positive. This signed threshold value is then subtracted from the residual gradient image. When applied to the whole movie, the event stream is then similar to the output of a neuromorphic camera [**?**], that is, a list of events defined by x_r and y_r (their position on the pixel grid), their polarity p_r (ON or OFF) and time t_r (see Fig. **??**-(c)). The goal here is to infer the correct motion solely by observing these events.

1.2 Hetero-synaptic delays model

The sensory signal representing the output of an event-based camera forms a discrete stream of events, which can be formalized as an ordered set of addresses and timestamps: $\epsilon = \{(a_r, t_r)\}_{r\in[1,N_{ev}]}$ where $N_{ev} \in \mathbb{N}$ is the total number of events in the data stream and the rank r is the index of each event ϵ_r. This event has a time of occurrence t_r and an associated address, which is typically in the form $a_r = (x_r, y_r, p_r)$. This defines a presynaptic address space $\mathcal{A} = [1, N_{\mathrm{X}}] \times [1, N_{\mathrm{Y}}] \times [1, N_{\mathrm{p}}] \subset \mathbb{N}^3$ where $(N_{\mathrm{X}}, N_{\mathrm{Y}})$ is the size of the sensor in pixels and N_{p} is the number of polarities ($N_{\mathrm{p}} = 2$ for ON and OFF polarities).

We may now define a layer of neurons $n \in \mathcal{B}$ by first describing how each neuron connects to presynaptic afferent from $\mathcal{A}$. In biology, a single cortical neuron has generally several thousands of synapses, and each synapse may be defined by its synaptic weight and its delay, that is, the time it takes for one spike to travel from the presynaptic neuron's soma to that of the postsynaptic neuron. A postsynaptic neuron $n \in \mathcal{B}$ is then not only described by synaptic weights connecting to a presynaptic afferent from $\mathcal{A}$ but also by the set of possible delays. For each neuron n, we define a set of N_s^n synapses, as $\sigma^n = \{(a_s^n, w_s^n, \delta_s^n)\}_{s\in[1,N_s^n]}$, where each synapse σ_s^n is associated to a weight w_s^n, a delay δ_s^n and a presynaptic address a_s^n. Note that a neuron may contact an afferent neuron with multiple different delays.

The corresponding input presynaptic spikes ϵ will be integrated by this synaptic set and notably by the respective delays, which will multiplex in time all possible patterns. For each time t the integration of ϵ is defined by a list of weights $\mathcal{W}^n$ linked to the synapses that match a precise spatio-temporal motif as input: *toto*. The activation function of our spiking neuron is a softmax function implementing a form of Multinomial Logistic Regression (MLR) [**?**], in analogy to a spiking Winner-Take-All network [**?**]. It transforms this list of weights into a probability with the following formula: $toto\frac{1}{Z}\exp(\mathcal{C}^n(t) + b^n)$ where $\mathcal{C}^n(t) = \sum \mathcal{W}^n(t)$ is the sum of the synaptic weights and b^n is the bias linked to neuron n. In particular, we expect that some specific motifs may become tightly synchronized as they reach the basal dendritic tree, leading to a high postsynaptic activity which makes it progressively more likely to generate an output spike.

1.3 Temporal Logistic Regression

In our MLR model with N_{class} classes, a probability value is predicted for each event at address a_r and at time t_r as a softmax function described in the above hetero-synpatic delays model. Such a probability can be computed for each neuron n in association to a specific class. From the perspective of simulating such event-based computations on standard chips, it is useful to transform this sparse representation into a dense representation. As such, we may first write any event-based input as the boolean matrix $A \in \{0, 1\}^{\mathcal{A}}$. In this simplified model, we will consider that hetero-synaptic delays are limited in range such that the synaptic set can be represented by the dense matrix K^n giving for each neuron n the nonzero weights as a function of presynaptic address and delay: $\forall s \in [1, N_s^n], K^n(a_s^n, \delta_s^n) = w_s^n$. Using this

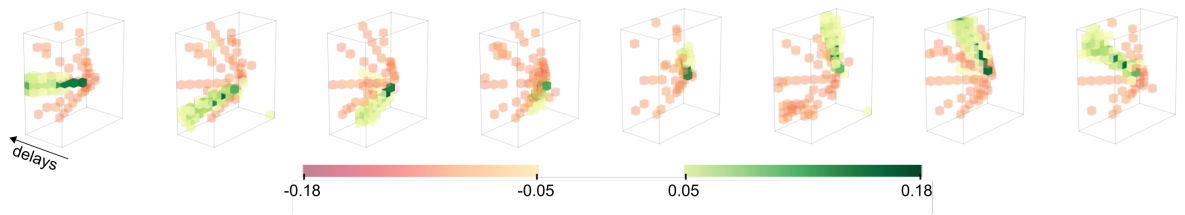

Fig. 2: Representation of the weights for the 8 learned kernels of the model corresponding to the OFF polarities and selective to the different motion directions (because of the symmetry observed between the ON and OFF event streams, kernels are similar for the ON polarities). These weights are associated to a specific delay on the *delays* axis and to a presynaptic address defined on the two other axes. For the sake of clarity, the values in range $[-0.05, 0.05]$ are not shown. One sees positive (excitatory) coefficients for the specific direction of motion and negative (inhibitory) coefficients for all other directions.

dense representation, the counting defined above becomes:

$$\mathcal{C}^n(a,t) = \sum_{a,\delta_s^n} K^n(a_s^n, \delta_s^n) \cdot A(a, t - \delta_s^n)$$

This shows that $\mathcal{C}^n$ is a temporal convolution of the dense representation of the event stream with the dense kernels formed by the set of synapses: $\mathcal{C}^n = K^n * A$. This well-known computation defines a time-invariant, differentiable measure which is very efficiently implemented for GPUs and which we will use for learning the classification of different patterns in the event stream. In particular, we may extend the convolution to a 3D convolution such that the representation would also benefit from spatial invariance. The use of spatio-temporal filters on a stream of events was shown to improve CNN performances for an action recognition task in [**?**]. For that, we design 3D kernels of shape $(K_\mathrm{x}, K_\mathrm{y}, K_\mathrm{t}) = (15, 15, 8)$, respectively representing the two spatial dimensions and the range of delays. An example of such kernels is given in Figure **??**. Computations are performed on spatio-temporal windows, defined by the kernels, sliding around the events, that is, the center of the spatio-temporal window around the current event ϵ_r. Finally, the output of the MLR model results in an event with the highest probability class, keeping the same timing as the event as input. The loss function of the MLR model is the binary cross entropy on the output of the classification layer. Simulations are performed thanks to the PyTorch library using gradient descent with Adam (with 2^{12} epochs and a learning rate of 10^{-5}).

2 Results

2.1 Kernels learned for motion detection

After training our model, we first observe the weights learned for the different neurons (see Fig. **??**). Focusing on the positive weights, a strong selectivity is observed along specific axes for the different kernels. These directions can be easily associated to the direction of motion controlled in the motion clouds. For instance, the third and the seventh kernels show a horizontal selectivity to motion directions. With the negative weights, one can observe an anti-selectivity for directions that do not correspond to the motion to which the kernel is selective to. This qualitative look at the 3D kernels allows the reader to infer for the 8 different motion directions used to generate our synthetic event streams.

If one focuses on the interpretation of these kernels in terms of spatio-temporal patterns embedded in the event stream, it can lead to interesting outcomes. In [**?**], a link between event-based MLR training and Hebbian learning is drawn, allowing to say that the present model will learn its weights according to a presynaptic activity associated to the different motion directions. Each neuron becomes selective to a specific motion direction through the learning of an associated prototypical spatio-temporal spike pattern. Each voxel in the 3D kernels defines a specific timestamp and a specific address. Then, our model is able to detect precise spatio-temporal patterns embedded in the spike train and associated to the different motion directions. The cone shape for the

positive weights distribution highlights a loss of precision for longer delays, i.e. events away in the past. For the directions not coherent to the class of a training sample, an anti-Hebbian learning is also observed through the negative weights in the kernels of Figure **??**.

2.2 Accuracy for the motion detection task

Once our MLR is trained, we obtain 3D kernels corresponding to the weights associated to the hetero-synaptic delays of our layer of spiking neurons and which may be used for detection. We observed that the distribution of the kernels' weights is sparse, with most values near zero. As shown in the formalization of our event-based model, the computational cost of our model if implemented on a neuromorphic chip would be dominated by the number of spikes times the number of synapses. This scales with the number of nonzero synaptic weights. To assess the robustness of the classification as a function of the computational load, we will prune the weights in $\{\sigma_s\}_{s\in[0,N_s)}$ that are below a defined threshold.

In Figure **??**, we plot the classification accuracy as a function of the relative number of computations, or active weights, per decision for each neuron of the layer. As a comparison and to account for the gain in performance by using hetero-synaptic delays, we provide the accuracy obtained with a MLR model using 2D time surface (in red) as in [**?**]. This latter method is based on delays from the last recorded events and uses fewer computations (in our case 15×15) than the dense 3D kernels without any pruning ($15\times15\times8$). While less computations are needed, the classification performance obtained for the model using time surfaces is similar to our method using all the weights of the kernels. By pruning weights, we observe that the evolution of accuracy as a function of the log percentage of active weights follows a sigmoid curve. Half-saturation level is reached at about 3.5×10^{-3}% of active weights, corresponding to a total amount of 6 computations per decision. Compared to the full kernels, the accuracy of our method is maintained to its top performances when dividing the number of computations by a factor up to about 200. In this case, the number of computations is greatly reduced compared to [**?**], thus demonstrating the efficiency of the presented method.

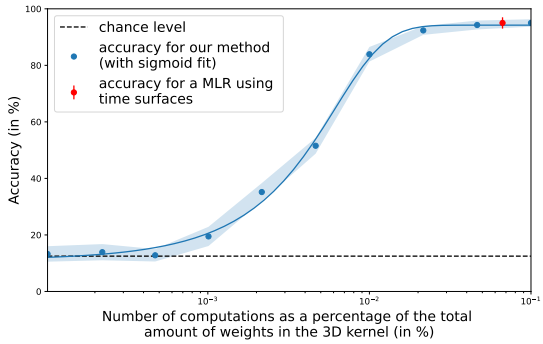

Fig. 3: Accuracy as a function of the number of computation load for the hetero-synaptic delays model (in blue) and for a method using 2D time surfaces (in red). The relative computational load (on a log axis) is controlled by changing the percentage of active weights relative to the dense convolution kernel. We observe a similar accuracy than HOTS, yet that our model could achieve a similar accuracy with significantly fewer coefficients.

3 Discussion

Here, we have introduced a generic SNN using hetero-synaptic delays and shown how it compares favorably for a visual motion detection task with a state-of-the-art event-based algorithm used for classification. The event-driven computations of our method can be reduced drastically through the pruning of synapses, while maintaining top performance for classification. This shows that we may use the precise timing of spikes to enhance neural computations.

Note that this supervised learning scheme can be extended to a variety of tasks. It would follow from the emergence of new kernels adapted to this new task after supervised learning. This constitutes a major advantage over other algorithms which derive event-based algorithms from specific physical rules (see for instance [**?**] for computing the optic flow using the luminance conservation rule). We aim at extending the application of this model on more generic datasets acquired in

natural conditions for progressively more complex tasks from motion extraction, optic flow or time-to-contact maps.