Performance comparison of DVS data spatial downscaling methods using Spiking Neural Networks

Amélie Gruel, Jean Martinet, Bernabé Linares-Barranco, Teresa Serrano-Gotarredona

Performance comparison of DVS data spatial downscaling methods using Spiking Neural Networks

Amélie Gruel
CNRS, i3S, Université Côte d'Azur
Sophi-Antipolis, France
amelie.gruel@univ-cotedazur.fr

Jean Martinet
CNRS, i3S, Université Côte d'Azur
Sophi-Antipolis, France
jean.martinet@univ-cotedazur.fr

Bernabé Linares-Barranco
IMSE-CNM
Sevilla, Spain
bernabe@imse-cnm.csic.es

Teresa Serrano-Gotarredona
IMSE-CNM
Sevilla, Spain
terese@imse-cnm.csic.es

Abstract

Dynamic Vision Sensors (DVS) are an unconventional type of camera that produces sparse and asynchronous event data, which has recently led to a strong increase in its use for computer vision tasks namely in robotics. Embedded systems face limitations in terms of energy resources, memory, computational power, and communication bandwidth. Hence, this application calls for a way to reduce the amount of data to be processed while keeping the relevant information for the task at hand. We thus believe that a formal definition of event data reduction methods will provide a step further towards sparse data processing.

The contributions of this paper are twofold: we introduce two complementary neuromorphic methods based on Spiking Neural Networks for DVS data spatial reduction, which is to best of our knowledge the first proposal of neuromorphic event data reduction; then we study for each method the trade-off between the amount of information kept after reduction, the performance of gesture classification after reduction and their capacity to handle events in real time. We demonstrate here that the proposed SNN-based methods outperform existing methods in a classification task for most dividing factors and are significantly better at handling data in real time, and make therefore the optimal choice for fully-integrated energy-efficient event data reduction running dynamically on a neuromorphic platform. Our code is publicly available online at: `https://github.com/amygruel/EvVisu`.

1. Introduction

The joint use of silicon retinas (Dynamic Vision Sensors, DVS) and Spiking Neural Networks (SNNs) is a promising combination for dynamic visual data processing. Both technologies have recently emerged separately about a decade ago from electronics and neuroscience communities, sharing many features: biological inspiration, temporal dimension, model sparsity, aim for a higher energy efficiency, etc.

However, deep SNN models can have difficulties handling a great amount of data simultaneously while minimising its energy consumption, especially at a high temporal resolution. One way to reduce their spiking activity while maintaining a high performance is to compress these models using attention, as proposed in [14]. Another way simply is to reduce the data these models use as input, meaning in our case reducing the event stream.

Furthermore, event cameras have a very high temporal resolution, in the order of microseconds. This means that theoretically, one pixel can generate at most 1 million events per second; thus a DVS128 of resolution 128×128 could produce 16.4 billion events per second. While this case never happens in real use due to the events' intrinsic sparsity and non-redundancy, several million events per second have been recorded in highly dynamic and textures scenes [24]. DVS sensors can thus produce heavy event streams requiring significant resources to handle. When implemented onto embedded systems, the processing can suffer from hardware limitations such as energy resources, memory, computational power, and communication bandwidth. In such situations where a dense event stream can neither be stored nor processed online, it is likely that some (if not most) events will be randomly dropped, which will obviously impede a correct processing.

It is thus necessary to design a sound way to filter events. We believe a straight-forward way to reach this goal is reducing the amount of input data received by the embedded system by artificially downscaling the size of the visual sensor. To the best of our knowledge, this problematic has been discussed only twice. Firstly, the Python library for neuromorphic data processing *Tonic* [16] implements a tool for spatial event downscaling. Secondly, authors in [12] have elaborated several innovative methods for event reduction.

In this paper, our core objective lies in designing a fully-integrated energy-efficient SNN model able to run dynamically on a neuromorphic platform. We claim the importance of providing downscaling methods for event data in order to filter out events and reduce the data flow with care, as this preprocessing step should be achieved while preserving the information conveyed. We introduce and benchmark several methods for spatially downscaling event data with SNNs. The design of such a model is not straightforward, and it requires some hyper-parameter tuning to achieve the results presented in the paper. As far as we know, it is the first time an SNN has been studied as a method to downscale event based data. Knowing that the combination of SNNs with event-based cameras seems to prove its worth in terms of compatibility of data types [3], information retention [13], energy efficiency [3, 7] and processing latency [7] reinforces the relevance of studying event data reduction using SNNs.

2. Event cameras

Conventional video cameras are based on the periodic acquisition of frames. Each frame is a static representation of the visual scene that is acquired by measuring the average light intensity during a certain sub-period time commonly known as exposure time. The intensity of each pixel is periodically measured (with a typical frame period of 20-30ms) and communicated regardless of whether it contains relevant information or not. Furthermore, objects moving at high speed relative to the frame time appear blurred in the image frame.

However, biological vision systems are not frame-based. In biological retinas, retinal pixels observe the visual scene and generate spikes whenever some relevant information is detected. These spikes are generated in a continuous and asynchronous way when a high spatial or temporal contrast is detected in the scene. The retinal spikes are sent through the optic nerve to be processed by the neural layers of the visual cortex. This spike based coding of the visual scene results in higher temporal resolution and lower communication and computational load of biological visual systems versus artificial ones.

DVSs implement a simplified retinal model where each pixel responds autonomously to the relative temporal variations of the illuminations [18]. These sensors have emerged

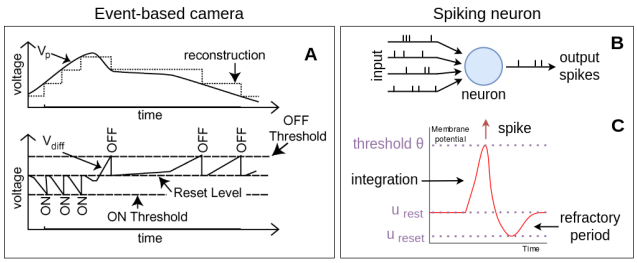

Figure 1: (A) Principle of operation of an event-based camera, from [18]. (B) Behavior of a spiking neuron, which receives spike trains as input and processes this information to produce a new sequence of activations. (C) Evolution of the neuron's membrane potential over time when activated by input spikes.

as the first commercially available neuromorphic sensors and have overcome the integration density and mismatch limitation exhibited by previous neuromorphic retinas.

Fig.1A illustrates the operation principle of a DVS pixel. As can be observed, DVSs generate sparse data. Each DVS pixel generates a string of spikes which respond to the temporal variation of the illumination. Consequently, the pixel remains silent under a static background saving communication and computation energy of the subsequent recognition network. Despite the sparse nature of data generated by DVS sensors, the increasing resolution of DVS prototypes [2] has motivated the research on active reduction methods for DVS data to save communication and computation bandwidth and energy.

3. Spiking Neural Networks

SNNs [21] represent an asynchronous type of artificial neural network closer to biology than traditional artificial networks, mainly because they seek to mimic the dynamics of neural membrane and action potentials over time. SNNs receive and process information in the form of spike trains, meaning as a non-monotonous sequence of activations, as represented in Fig. 1AB. Therefore, they make for a suitable candidate for the efficient processing and classification of incoming event patterns measured by silicon retinas. This spatio-temporal model allows capturing and processing the dynamics of a scene. Moreover, since this model is sparse, it enables energy efficient implementations.

A SNN is constructed using populations of neurons linked together with connections, according to certain rules and a certain architecture. By definition, a spiking neuron follows a model based on parameters describing its internal state and its reaction to the input current (as pictured in Fig. 1B). Many models exist; from this set we chose to use the Leaky Integrate-and-Fire (LIF) model within the Spiking Neural Network Pooling method.

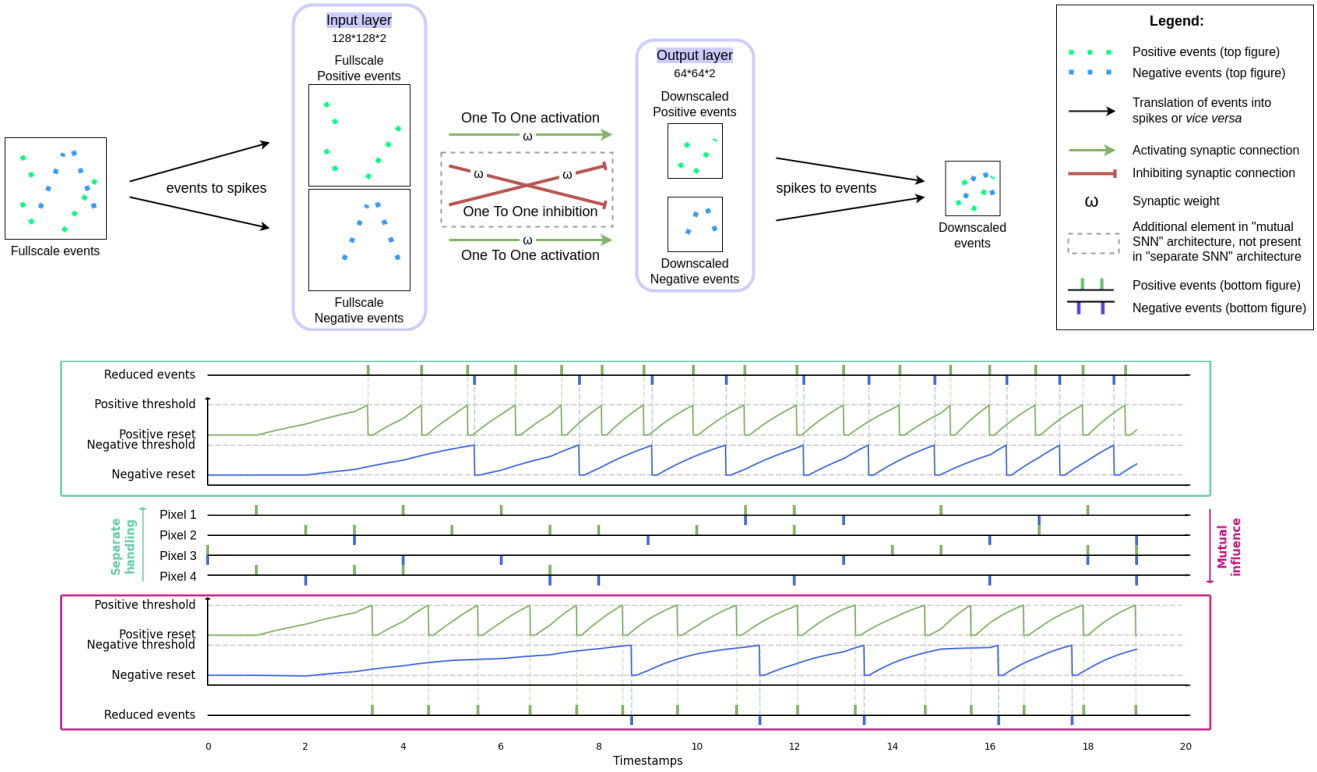

Figure 2: **Above** The two architectures proposed by the authors for SNN Pooling either handle the positive and negative events separately (excluding the grey dotted box in the figure) or take into account their mutual influence (including the grey dotted box in the figure). The green arrows represents All-To-All excitatory connections between an input sub-region and the corresponding output neuron of same polarity, and the red arrows All-To-All inhibitory connections between different polarities. The green dots corresponds to positive events, the blue dots to negative events. In this example those two architectures are applied to data recorded with a sensor size of 128 pixels, which are downscaled by 2. The output layer of each polarity is thus of size $(128/2)^2 = 64^2$.
Below Schematic illustration for SNN pooling. A sensor of size 2×2 outputs events both negative (blue ticks) and positive (green ticks) at each pixel across time, which are downscaled with SNN pooling handling separately the events (green frame on top) or allowing a mutual influence (pink frame at the bottom). In this example, the downscaling factor is set to 2. Best seen in colours.

Why should we use SNNs to process event data? SNNs are a sparse and dynamic model for processing spatio-temporal data. They are particularly well suited to handle the atypical kind of data output from event cameras, since each event can be assimilated to an activation spike between two spiking neurons. Furthermore, the behaviour of the log-luminance recorded by the sensor displays a strong similarity with one of a LIF neuron, as shown in Fig. 1. As of today, SNN simulations with a CPU require a substantial simulation time. However, the use of neuromorphic hardware such as SpiNNaker [10], BrainScaleS [22], or Loihi [4] enables fast and low power simulations.

4. Event downscaling by Spiking Neural Network pooling

The following subsections will present our contribution: two novel SNN architectures spatially downscaling event data by SNN pooling. Although such architectures are not innovative *per se*, its use in the context of event-driven data reduction is unprecedented.

4.1. How to downscale events using SNN?

In some ways, this event processing has often been used by SNN models handling neuromorphic data: indeed the use of a strided convolutions or max pooling layer e.g. in [3] is close to event downscaling by SNN pooling, as we present here. The events are directly translated as input spikes in a 2D layer of size $width \times height$ connected to a smaller 2D layer of size $(width/ratio) \times (height/ratio)$, which implements a convolutional layer with a kernel size $ratio \times ratio$, a stride $ratio$, without padding. Each spike output by this smaller layer is then interpreted as events, with each neuron construed as pixel of a smaller sensor. As described in the following paragraphs, we implemented

two SNN architectures depending on whether the positive and negative events are handled separately or not ; in other words whether the event data is downscaled as a whole or per polarity, with no mutual influence.

4.2. Separate handling of polarity

A first method of SNN pooling is proposed, where the positive and negative events are handled differently, although during the same simulation. The architecture used is depicted in Fig. 2 (top): the 2D input layer is of size $width \times height \times 2$ with two 2D set of LIF neurons stack together, each one receiving either positive or negative events at the corresponding neurons. Likewise, the output layer is of size $(width/ratio) \times (height/ratio) \times 2$ and each of its LIF neuron is wired to the corresponding input region by an excitatory connection. No learning is necessary here: the weight of the connection between the input and the output is set (as is the sensitivity of an event camera recording a scene - see below) and should not be adapted along the simulation. Spatially downscaling using SNN pooling relies entirely on the LIF neuron's intrinsic dynamics, which come close to the behaviour of a DVS pixel.

Most SNN taking events directly as input, with no intermediate preprocessing of the data, translate each of them into a spike at the corresponding coordinates in the input 2D layer of the network with no distinction between positive and negative events; any time a pixel activates, a spike is emitted by the corresponding input neuron (see [3]). This may not have an impact on some computer vision tasks, but in our case we aim to reduce event data while keeping the information as accurate as possible, thus preserving polarity. From this stems our choice for the particular architecture described above.

However, separately handling the event polarities disregards the physics behind event cameras and embodies the main flaw of this first architecture. In case of a burst of negative and positive events grouped together, the luminance is actually quite stable and varies around one value with a variance slightly higher than the sensor's sensitivity threshold. As can be seen in Fig. 2 (bottom), downscaling this data should lead to a smaller luminance variance, thus to less events produced since they compensate each other.

4.3. Mutual influence of positive and negative events

To resolve the aforementioned problem posed by a separate processing of polarity, we implemented a second architecture illustrated in Fig. 2 (top – including grey dotted box). It builds on the structure of the first one and adds to it a mechanism of mutual inhibition between the polarity. Each sub-region of the input layer activates the corresponding output neuron of the same polarity, but it also inhibits the corresponding output neuron of the opposite polarity with the same weight. This way each activation of the output neuron's membrane potential due to an event of a certain polarity is offset by any potential event of the opposed polarity taking place in a close time span., to obtain the expected behaviour presented in Fig. 2 (bottom).

This represents more faithfully the behaviour of an event camera of smaller resolution, recording the same luminance as the sample being downscaled. It should also be noted that the sensitivity of this simulated recording of a downscaled luminance can be adapted simply by adjusting the weight of the connections, as easily as adjusting the sensitivity of a real DVS: the higher the weights, the higher the activation of the output neurons and the more events produced, and reciprocally.

Both methods described above will respectively be referenced as "separate SNN" and "mutual SNN" methods in the following pages.

4.4. Influence of hyperparameter tuning on SNN pooling

As with most SNNs, hyperparameter tuning greatly influences the activation of the different layers and the overall spiking dynamic. Hyperparameters include the synaptic weight, which can excite the layer to varying degrees; the neurons' threshold, which makes them more or less likely to be activated depending on its value; the synaptic time constant, which corresponds to the time required for the membrane potential to decay to its reset value; etc. This hyperparameter tuning is not inconsistent with the physical operation set up of the DVS: indeed in a DVS camera different parameters can be set to record a scene, such as the contrast threshold or the refractory period.

It was thus necessary to study the physical properties of both separate and mutual architectures described in the previous section with varying hyperparameter tuning, in order to select the most optimal one. As seen earlier, many hyperparameters can be tuned in order to obtain the ideal results. Here, we limit our study to the synaptic weight ω (see Fig. 2 above) as it can be assimilated to a DVS contrast threshold. The rest of the hyperparameters are set to values commonly used to initialise a LIF neuron in a SNN (presented in Tab. 1). As pictured in Fig. 3, the higher the synaptic weight, the more events the connections allows through and

Parameter	value
Resting membrane potential	-65 mV
Reset membrane potential	-65 mV
Neuronal threshold	-50 mV
Membrane time constant	20 ms
Refractory period	0.1 ms
Excitatory decay time	5 ms
Inhibitory decay time	5 ms

Table 1: Hyperparameters used to initialise the separate and mutual SNN downscaling methods

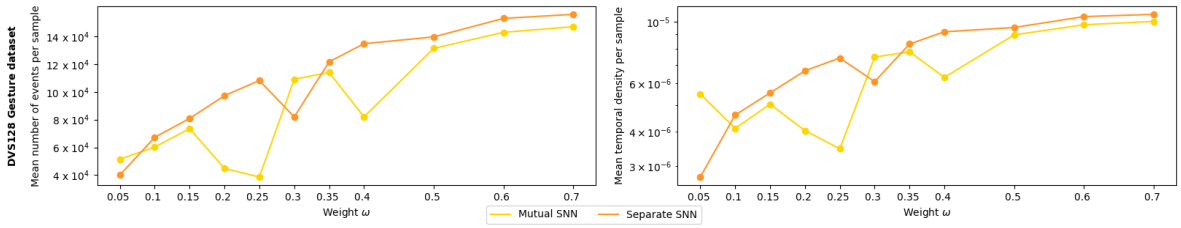

Figure 3: Comparison of physical properties according to a varying synaptic weight ω for separate and mutual SNN downscaling methods applied on DVS128 Gesture.

the higher the event density. If we consider the standard deviation of the density per pixel as a measure of information provided by the event data, we could set a hyperparameter threshold which once reached, produces non suitable downscaled data. Conversely, a lower bound should also be provided in order to keep enough events in the output to have a minimum of information. This is coherent with the contrast threshold used to set up a DVS camera: the higher the threshold, the more events are filtered and the less understandable the event data becomes. Hyperparameter tuning is therefore an important aspect to keep in mind when using SNNs to downscale events.

Fig. 3 also interestingly shows that the mutual SNN tends to output fewer events in a lower temporal density than separate SNN, which will be confirmed in Fig. 4. Indeed the mutual inhibition tends to normalise the membrane potential of each neuron (see Fig. 2 bottom), which can be assimilated to the log luminance recorded by each sensor's pixel.

4.5. Network implementation

Both methods were implemented in this work using the Python SNN simulator PyNN [6] to produce downscaled datasets for further comparisons. However, since this library runs simulations on CPU, it leads to a significant run time - we thus adapted the networks to be run on the SpiNNaker neuromorphic chip [9].

SpiNNaker is a neuromorphic hardware platform developed in the University of Manchester under the Flagship Human Brain Project. This microelectronical hardware communicates and process spike events. Two versions of SpiNNaker boards are available. The SpiNN-3 boards allocate 4 SpiNNaker chips thus can implement architectures with up to 18K neurons. Newly available SpiNN-5 boards contain 48 chips, having capacity for 195K neurons. Both were used in this work to run our network.

5. Comparison between event downscaling methods according to physical properties

We formalised previously two SNN architectures which spatially downscale event data. The following sections will now compare these two novel spatial event downscaling approaches to those formalised in [12], i.e. a simple event funnelling, a tally of positive and negative events and a log-luminance reconstruction (linear and cubic).

5.1. Event-based datasets

We describe below the neuromorphic datasets used in this work to benchmark these different spatial event downscaling methods. We selected both datasets because they are different but complementary in many aspects (sensor size, amount of events, visual information, etc).

The DVS128 Gesture dataset [1] has now become a standard benchmark in event data classification. It features 29 subjects recorded with a DVS128 camera, performing 11 different hand gestures under different illumination conditions. The *Neuromorphic-MNIST* dataset is an event translation of the original *MNIST* dataset [15] adapted to the neuromorphic by Orchard et al. [19]. N-MNIST is particularly heavy in events: according to the data presented in the above paragraphs, N-MNIST displays on average 10.71 events per pixel per second, whereas DVS128 Gesture only displays about 4.06 events per pixel per second.

5.2. Comparison between methods

The different event spatial downscaling methods described in [12] as well as in the section above vary strongly by their physical properties, as seen in Fig.4. The temporal density corresponds to the activation probability of pixels averaged over the whole sensor. The number of events and the temporal density have been averaged per sample.

The spatial downscaling of event data aims to reduce significantly the number of events per sample, while keeping a sufficient amount of relevant information.

Fig. 4 (left) show quite clearly that the number of events per sample decreases to a varying extent depending on the downscaling method used. The least convincing method by this criterion is obviously the funnelling one: the number of events stays stable and significantly high no matter the dividing factor, whereas the most pronounced drop is obtained with the log luminance reconstruction methods.

Fig. 4 (right) represents the temporal event density as the number of events is not enough a criterion to decide

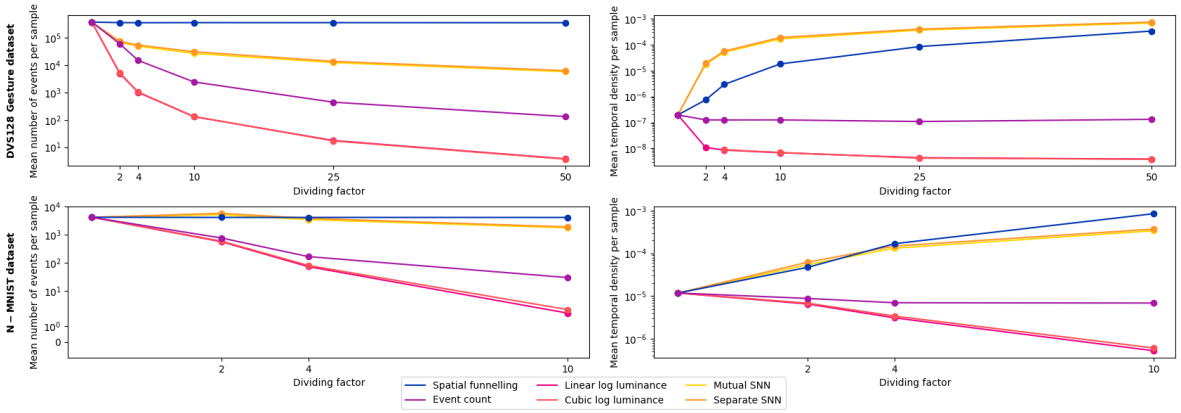

Figure 4: Comparison of physical properties between different methods and dividers for DVS128 Gesture (above) and NMNIST (below) datasets. We see here that the funnelling method keeps a high number of events and has an increasing temporal density, whereas all the other methods decrease strongly the number of events kept. The log luminance reconstructions lead to the decrease of temporal density, while the event count maintains one similar to the original data.

which method provides the most relevant information with an increasing dividing factor and further material needs to be bought. This criteria shows the distribution of the reduced event over the sensor: a high standard density corresponds to event data with a high activation. Fig. 4 shows that the funnelling and event count methods have a high temporal density, i.e. all the sensor's pixels tend to activate a lot across time thus reducing the information transmitted. *A contrario*, the log luminance reconstruction seems to provide comprehensive data, and both SNN methods seem a good compromise between these two *extrema*.

Since the spatial downscaling methods are more likely to be applied on an embedded system, it is interesting to consider the energy consumed by each method. The funnelling method reduces data with zero energy, as the funnelling can be applied in the event transmission from the sensor to the processor by just ignoring some bits in the address of the events. On an embedded computer, the log luminance reconstruction methods are made thanks to an algorithm of higher complexity thus requiring a greater energy expenditure. However if we consider this reduction at the sensor level, it can be produced at zero energy cost. Finally, the SNN methods would be directly performed by an embedded neuromorphic chip, whose energy consumption is directly proportional to the number of events given as input and the number of synapses implemented between the input and output layer, i.e. proportional to the dividing factor.

6. Study of real time spatial event downscaling

In this section we propose to compare the SNN pooling method introduced in this paper with the spatial downscaling methods described in [12] regarding real time data handling. Indeed, while event sensors show a number of advantages over frame-based cameras such as high temporal resolution, low power consumption, high dynamic range sensing, etc, that make them unavoidable for the future of real-time embedded vision systems, the limited processing resources make it often hard to deal with a high event density in real-time embedded vision applications. This dilemma calls for data reduction techniques able to handle input event in real time.

In order to investigate the relevance of the SNN downscaling method for real-time event handling, we implemented both SNN downscaling methods on the SpiNN-3 board (see above). The SpiNN-3 board is composed of 4 chips, each composed of 18 cores. Since each core can simulate up to 256 spiking neurons, the SpiNN-3 can simulate around 18,000 neurons at most. Thus the run time of the SNN spatial downscaling method suffers from the limitations of the SpiNN-3 board it is run on. The number of input events per simulation time-step does not influence the simulation run time, contrary to the number of neurons and connections. The proposed architecture is convolutional , meaning that the number of connections c depends on the number of neurons $n_{input} = 2 \times width \times height$ and $n_{output} = \frac{n_{input}}{ratio^2}$ in both layers, and the convolutional kernel size $kernel$, which is actually the same as $ratio$ in the proposed topology: $kernel = ratio$.

Interestingly, for given $width$ and $height$, the number of connections $c = kernel^2 \times n_{output} = n_{input}$ does not depend on $ratio$, whereas $ratio$ strongly influences the number of output neurons to be simulated on the board. Technically, PyNN uses a *SpikeSourceArray* (resp. a *SpikeInjector*) data structure for off-line (resp. on-line) neurons, which do not need to be simulated by the SpiNN-3 board. This important detail means that although the architecture requires $n_{input} + n_{output}$ neurons, the board will only need to simulate n_{output} neurons and their c connections. Depending on whether the separate or the mutual SNN downscaling version is used, the number of connections varies;

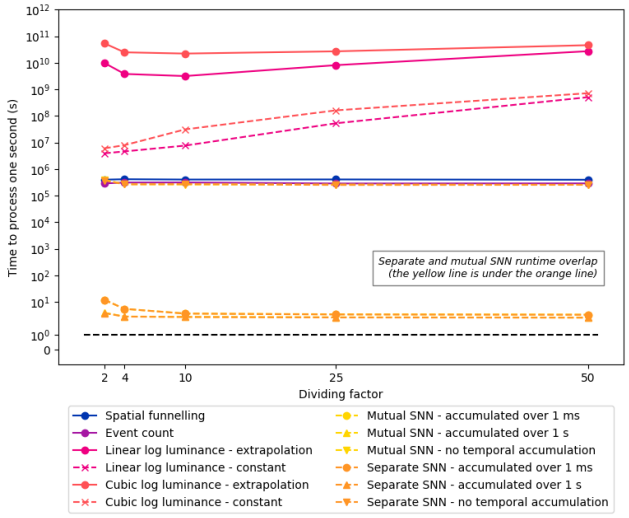

Figure 5: Run time evolution for spatially downscaling 1s of real time event data. The dashed line shows the real time bound. The SNN methods are undoubtedly the quickest, but still two order of magnitude slower than real time.

indeed the mutual method adds an inhibitory connection between neurons corresponding to positive events in input to those corresponding to negative events in output, and reciprocally. Thus the number of connections c doubles between the separate and mutual SNN methods. Considering all this information, the simulation run time should decrease when the dividing factor $ratio$ increases for both SNN downscaling methods.

The downscaling method using log-luminance reconstruction needs the information of when in the future will be the next event, therefore it requires an adaptation. The proposed adaptations describe the behavior of the reconstructed log-luminance curves when no input event is given at a time t: (1) the reconstructed curve remains constant until new events are triggered, (2) the reconstructed curve is extrapolated (linear/cubic) given the slope at the location of last triggered event. These two strategies will be respectively referenced as *constant* and *interpolation* in the remainder.

The real time spatial event downscaling studied in this section was run on Intel 8-core i9-10885H CPU at 2.4GHz. Fig. 5 illustrates the computational run time evolution. It should be noted that this CPU can only perform a maximum of 2.4 billion cycles per second, whereas an event camera can output up to 1 million events per second per pixel. For the DVS 128 Gesture dataset, the event camera is of size 128×128 pixels, thus can output up to 16 billion events per second (with an average of 30,000 events per second for this dataset, according to [12]). The speed of our CPU is thus too low to simply cycle through the events in real time, so real time reduction is unattainable using this hardware.

7. Study of spatial event downscaling influence on a classification task

Ideally an optimal spatial downscaling method for event data reduces drastically the number of events while conserving relevant data. This paper aims to discuss the relevancy of our proposed approaches compared to existing ones regarding this query. Physical properties were studied above as a first attempt to answer this interrogation. We further investigated the relevance of the retained information by looking at the performance of each method against a specific computer vision task, a classification.

7.1. Classifiers

We use to this end a classifier which take as input the data directly as individual events.

Shrestha and Orchard developed in 2018 a Python framework called SLAYER (for Spike Layer Error Reassignment in Time) [23], based on the machine learning library PyTorch [20] and designed to simulate "backpropagation based SNN learning" on GPU. A specific "SLAYER Loihi" module has been implemented to run SNN models initially developed on SLAYER, on Intel's Loihi neuromorphic chips [5] without further adaptation to the code. This module and more globally this framework were benchmarked on a classification task applied to different spiking and non spiking visual datasets, such as MNIST [15], N-MNIST [19] and DVS128 Gesture [1], and the audio dataset TIDIGITS converted into spikes [17]. According to the authors, when the input is a spiking dataset "the spike data from the DVS is directly fed into the classifier".

In order to optimise the simulation run time, we preloaded and fine tuned this classifier's weights in a transfer learning strategy on the downscaled datasets.

7.2. Performance evolution for different downscaling methods

We apply the SLAYER classifier on the DVS128 Gesture and N-MNIST datasets reduced using the two novel SNN approaches presented previously and the methods presented in [12]. The resulting performances are presented in Fig. 6. The optimal downscaling method should have the best accuracy performance, since it produces the most relevant information necessary for an effective classification.

Unlike other classification models which accumulate input events into frames (such as PLIF [8]), the SLAYER classifier takes as input sparse events and shows more interesting and coherent results than the previously mentioned would (see Supplementary Material). Fig. 6 shows that the downscaling methods introduced in this paper perform well, especially for the DVS 128 Gesture dataset. Regarding N-MNIST, as seen earlier, the large number of events retained for a small dividing factor depends entirely on the hyperpa-

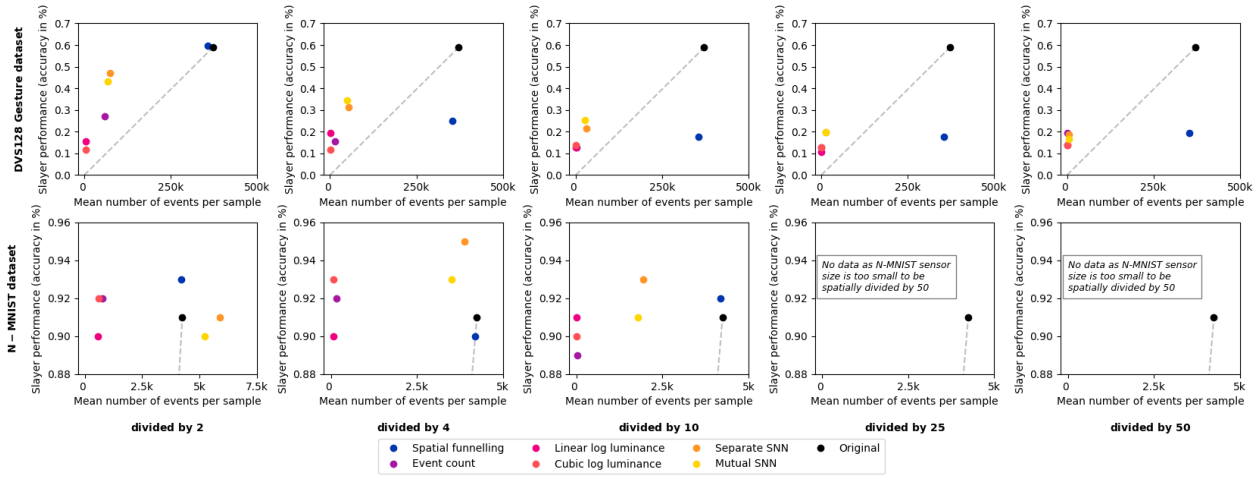

Figure 6: Evolution of the accuracy performance of the SLAYER classifier applied to DVS128 Gesture (first line) and N-MMNIST (second line). Note that the points located in the upper left side of the dashed line correspond to results showing interesting trade-offs for performance over data size, i.e. a high accuracy with small number of events. Globally, most spatial downscaling methods lead to a good trade-off, with the notable exception of the "Spatial funnelling" which remain to the lower right even when the dividing factor increases.

rameter tuning set up and could be tuned for each dataset. Moreover, this shortcoming is compensated by the significant advantage of fast computation time for SNN pooling compared to other methods (see Section 6). In contrast, the "Spatial funnelling" remains on the lower right even when the dividing factor increases ; moreover past a small dividing factor (2), its performance is lower and number of retained events higher than most methods. It should be noted that all methods lead to the decrease of performance according to the increase of the dividing factor, which validate the physical properties study performed in above section.

Conclusion

The spatial downscaling of event data on an embedded system aims to reduce significantly the number of events per sample, while keeping a sufficient amount of relevant information and handling event close to real time. In this work we formalise two novel spatial event downscaling approaches based on SNN pooling. The choice to use one of these rather than one of the methods presented in [12] is not trivial but depends strongly on the computational vision task we wish to accomplish, the application context, on the event data preprocess it requires and the embedded computer throughput.

The best choice to perform event reduction on an embedded system in order to adjust the complexity of data to the available resources such as processing capability and power consumption, can be found among the SNN pooling methods introduced in this paper. Indeed they achieve an efficient trade-off between the optimisation of desired physical properties, performance and energy consumption as well as being significantly closer to real time data handling than existing methods. Furthermore, when neuromorphic hardware is involved, it can be deemed easier and less costly to use either SNN methods described in this paper as a first preprocessing layer to machine learning models.

As no significant differences can be observed between the run time and the classification performance of mutual and separate SNN pooling, we look at the physical properties to decide between these two methods. The mutual SNN pooling is definitely the most optimal one based on the results of this work, as it leads to a smaller number of events and a lower temporal density and is more optically coherent with the behaviour of an event camera.

Further work on event based downscaling include the implementation of a event based foveation mechanism such as the one described in [11]. Another axe of research concerns event temporal downscaling, of which we have identified two main approaches. One consists in changing the grain of the timestamps, for example from nanoseconds to a bigger unit of time such as microsecond, millisecond, second... considering all events in one instance of this bigger unit as occurring at the same timestamp. Another one involves the selection of k events randomly trough-out the input data, or at a specific time interval. This second method could be more realist regarding the behaviour of a DVS connected to a model unable to process the input flow.

Acknowledgement

This work was supported by the European Union's ERA-NET CHIST-ERA 2018 research and innovation programme under grant agreement ANR-19-CHR3-0008.

The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support.

References

[1] A. Amir et al. A Low Power, Fully Event-Based Gesture Recognition System. In *CVPR*. IEEE, 2017.

[2] Shoushun Chen and Menghan Guo. Live demonstration: Celex-v: A 1m pixel multi-mode event-based sensor. In *CVPRW*, 2019.

[3] Loic Cordone, Benoit Miramond, and Sonia Ferrante. Learning from Event Cameras with Sparse Spiking Convolutional Neural Networks. In *IJCNN*, 2021.

[4] Mike Davies et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 2018.

[5] Mike Davies et al. Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109, 2021.

[6] Andrew P Davison, Daniel Brüderle, Jochen M Eppler, Jens Kremkow, Eilif Muller, Dejan Pecevski, Laurent Perrinet, and Pierre Yger. Pynn: a common interface for neuronal network simulators. *Frontiers in Neuroinformatics*, 0, Jan 2009.

[7] Guillaume Debat, Tushar Chauhan, Benoit R. Cottereau, Timothée Masquelier, Michel Paindavoine, and Robin Baures. Event-based trajectory prediction using spiking neural networks. *Frontiers in Computational Neuroscience*, 15, May 2021.

[8] Wei Fang et al. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *ICCV*, 2021.

[9] Steve Furber and Petrut Bogdan. *Spinnaker - a spiking neural network architecture*. NOW Publishers INC, 2020.

[10] Steve B. Furber et al. Overview of the spinnaker system architecture. *IEEE Transactions on Computers*, 2013.

[11] Amélie Gruel, Jean Martinet, Bernabé Linares-Barranco, and Teresa Serrano-Gotarredona. Stakes of foveation on event cameras. In *ORASIS 2021*, 2021.

[12] Amélie Gruel, Jean Martinet, Teresa Serrano-Gotarredona, and Bernabé Linares-Barranco. Event data downscaling for embedded computer vision. In *VISAPP*, 2022.

[13] Jesse Hagenaars, Federico Paredes-Valles, and Guido de Croon. Self-supervised learning of event-based optical flow with spiking neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 7167–7179. Curran Associates, Inc., 2021.

[14] Souvik Kundu, Gourav Datta, Massoud Pedram, and Peter A. Beerel. Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression. *WACV*, 2021.

[15] Yann Lecun, Leon Bottou, Y. Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 12 1998.

[16] Gregor Lenz et al. Tonic: event-based datasets and transformations., July 2021. Documentation available under https://tonic.readthedocs.io.

[17] R. Gary Leonard and George Doddington. Tidigits. Philadelphia: Linguistic Data Consortium, 1993.

[18] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128x128 120 db 15 us latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 2008.

[19] Garrick Orchard et al. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, 9:437, 2015.

[20] Adam Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

[21] Hélène Paugam-Moisy and Sander M. Bohte. Computing with Spiking Neuron Networks. In *Handbook of Natural Computing*. Springer-Verlag, Sept. 2012.

[22] Johannes Schemmel et al. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *ISCAS*, 2010.

[23] Sumit Bam Shrestha and Garrick Orchard. SLAYER: Spike layer error reassignment in time. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018.

[24] Mohammad-Hassan Tayarani-Najaran and Michael Schmuker. Event-based sensing and signal processing in the visual, auditory, and olfactory domain: A review. *Frontiers in Neural Circuits*, 0, Jan 2021.