Scene Context Classification with Event-Driven Spiking Deep Neural Networks

Pablo Negri*, Miguel Soto†, Bernabé Linares-Barranco† and Teresa Serrano-Gotarredona†
*Instituto en Ciencias de la Computación (UBA-CONICET), Buenos Aires, Argentine
pnegri@dc.uba.ar
† Instituto de Microelectrónica de Sevilla (IMSE-CNM), CSIC and Universidad de Sevilla, Sevilla, Spain

Abstract—**Event-Driven computation is attracting growing attention among researchers for several reasons. On one hand, the availability of new bio-inspired retina-like vision sensors that provide spiking outputs, like the Dynamic Vision Sensor (DVS) make it possible to demonstrate energy efficient and high-speed complex vision tasks. On the other hand, the emergence of abundant new nanoscale devices that operate as tunable two-terminal resistive elements, which when operated through dynamic pulsing techniques emulate learning and processing in the brain, promise an explosion of highly compact energy efficient neuromorphic event-driven applications. In this paper we focus for the first time on a high-level cognitive task, namely scene context classification, performed by event-driven computations and using real sensory data from a DVS camera.**

I. Introduction

Scene recognition targets the categorization of the environment where a particular view has been recorded. Knowledge about the scene may enable a more intelligent vision processing [1] and can provide valuable clues for navigation tasks and developing assisted technologies for visually impaired people that can make use of contextual information.

Previous works on scene recognition are based on images captured with conventional photograph or video cameras. Images are acquired as periodic frames. In the present work, scenes are captured from an unconventional event-driven dynamic vision sensor (DVS) [2], [3], [4], [5], [6]. Taking inspiration from biological systems, event-based sensing and recognition systems operate in a continuous and asynchronous way. In a DVS sensor, each pixel responds to temporal changes in the illumination, thus providing event-driven sensory data. Several advantages emerge from such event-driven computational paradigm. First of all, the communication and subsequent computation is performed only when there is a change or relevant information transmitted saving thus computation and communication power and bandwidth. Secondly, the computation is performed on the fly, event by event without waiting for full frames. Thus, recognition can be performed with a very reduced latency [7], [8]. Thirdly, event-driven neuromorphic computation fits naturally with emerging nanoscale devices that can learn and process through spikes (events), while resulting in highly compact, very low power hardware, but capable of high-level cognitive computations.

This paper tackles for the first time a problem which combines scene recognition with DVS information. The principal objective is to identify known places, or to characterize the content of a scene, in order to discriminate their particular structure. The input queries will consist of scenes recorded by a DVS carried by a walking person.

II. Scene Recognition System

A. Dataset

The dataset was recorded by different people carrying a DVS during hikes in different urban or in-door environments. The sequences exceeded 15 minutes duration and include indoor and outdoor locations. The recordings were split into small temporal sequences $\mathbf{e}_{\Delta t}$, with $\Delta t = 50$ msec, separated by 100 msec.

To categorize the sequences, we defined $J = 4$ classes: ***street***, ***bridge***, ***stairs***, and ***corridor***. Fig. 1 shows samples of the J classes as RGB captures in order to illustrate the views, and the DVS events from the places. Both ***street*** and ***bridge*** are outdoors views. The class ***street*** is composed of urban scenes, mainly captured from the sidewalk. This is the class with the highest intra-class diversity, which have a great variety of textures: buildings, trees, vehicles, etc. The ***bridge*** class records the view crossing a bridge, as can be seen in Fig. 1. The indoors views are represented by the classes ***stairs*** and ***corridor***. The samples of the former class always have a stair in their view. Fig. 1 illustrates with two different samples the diversity of views within this class. The ***corridor*** class captures the view of corridors inside buildings.

B. Event-Driven Processing Simulator: MegaSim

The data flow captured by the event-driven DVS feeds our scene classifier as a continuous stream of temporal asynchronous events. The scene classifier is a neural network architecture composed of several layers of interconnected event-driven neuron populations. In order to simulate its behavior, we employ a tool called MegaSim (Modular Event-Driven Growing Asynchronous Simulator) [9].

MegaSim, which is written in C, allows the analysis of arbitrary neural networks topologies described in a *netlist* text file. The topology is described as a network of modules and AER (Address Event Representation) nodes. In an AER node (bus) many inter-neuron connections are time-multiplexed [10]. Each module is connected to one or more input AER nodes and one or more output AER nodes. Each module represents a neural population with a particular behavior that can be defined by the user. Each AER node is represented by

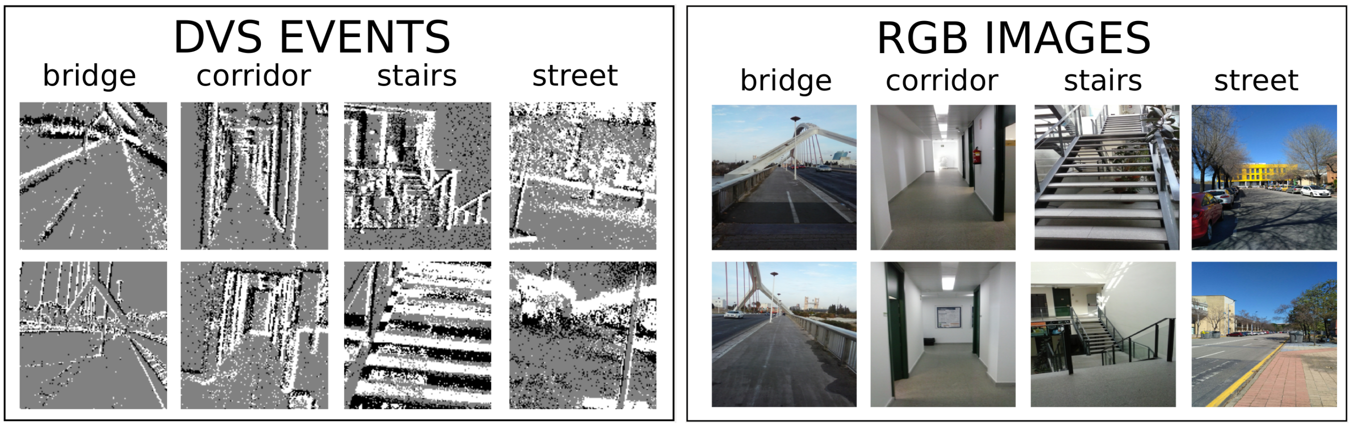

Fig. 1. (Left) Two dimensional representations of the event-based information provided by the DVS, captured by a walking pedestrian. DVS frames show the information accumulated during 50 msec. (Right) RGB images are shown to illustrate the appearance diversity within the classes.

a list of events generated along time by the neurons whose outputs are connected to that particular AER node.

Initially, the event lists in the network internal AER nodes are empty. Only the node corresponding to the DVS output is full with the events corresponding to the DVS recordings. As the simulation runs, the DVS event data are processed by the subsequent neural modules that generate new events that are progressively added to the corresponding output AER nodes. At each simulation time step, the simulator looks for the older unprocessed event in the network and calls the corresponding modules receiving it. Megasim is a powerful tool to simulate Spiking Neural Networks behavior, and to perform a rapid prototyping and testing before building hardware architectures, i.e. on FPGAs or ASICs.

C. *SNN Architectures*

The Spiking Neuron Network of our system employs a leaky integrate-and-fire neuron model (LIF), where the potential $v_j(t)$ of post-synaptic neuron j (also called *neuron state*) receives the contribution of presynaptic neurons via weighted synapses. In this simple form the neuron is modeled by:

$$\frac{\partial v_j(t)}{\partial t} = -\frac{v_j(t) + C}{\tau_l} \tag{1}$$

where C is a constant, and τ_l defines the leakage rate. If there is no input spike increasing or decreasing the neuron state, $v_j(t)$ is subject to leakage only. When $v_j(t)$ reaches a certain threshold T_{MAX}, the neuron generates a spike output event to the subsequent neurons in the network, and $v_j(t)$ gets instantaneously a reset value v_r.

We propose two topologies for the spiking neural network event-driven classifier.

The first proposed topology is a two-layer architecture as depicted in Fig. 2. This topology, proposed in [11], has a feature extraction (FE) phase composed by layer C1 followed by a subsampling process S1, a flattening process (recoding neuron addresses from 2D to 1D), and a fully connected output classification/prediction layer.

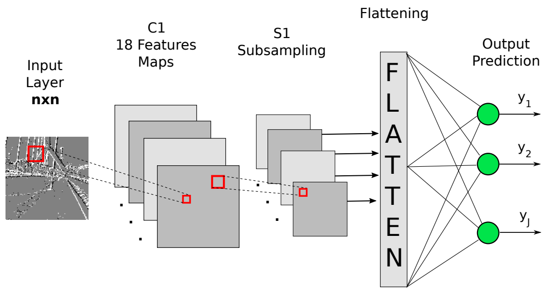

Fig. 2. Two-Layer Spiking Neural Network.

The spikes generated in the $n \times n$ DVS retina, with $n = 128$, feed the C1 convolutional Feature Maps with kernels of size $k \times k$, also called receptive fields. The kernels consist of 18 Gabor Filters with the following equations:

$$\begin{aligned} g(x, y, \lambda, \theta, \phi, \sigma) &= exp\left(-\frac{1}{2}\left(\frac{x'^2}{\sigma_x^2} + \frac{y'^2}{\sigma_y^2}\right)\right) \\ &\times cos\left(2\pi\frac{x'}{\lambda} + \phi\right) \end{aligned} \tag{2}$$

where $x' = x\cos\theta + y\sin\theta$, $y' = -x\sin\theta + y\cos\theta$, λ is the wavelength of the sinusoidal, θ is the orientation of the Gabor filter, ϕ is the phase offset, and σ is the Gaussian' width. The 18 kernels are obtained using 9 orientations and 2 phase values[1]. We use $k = 7$ in this paper. The total number of neurons in C1 is $18 \times ((n - k + 1) \times (n - k + 1))$.

The subsampling process S1 applies a factor 2 down sampling. The output of S1 is rearranged into a one-dimensional vector by module "Flattening" in Fig. 2. Its size is $N = 18 \times (((n - k + 1)/2) \times ((n - k + 1)/2))$. The flattening module output is fully connected to the output classifier which consist of J neurons. Pre-synaptic spikes from neuron i in the flattening module contribute to the voltage membrane of each

[1]The value of these parameters are; $\theta = [0, 20, 40, 60, 80, 100, 120, 140, 160]$, $\sigma = 4$, $\lambda = 8$, $\phi = [0.0, 1.7]$

output neuron j by decrementing or incrementing their state v_j proportionally to the corresponding synaptic weights $|w_{ij}^{(1)}|$.

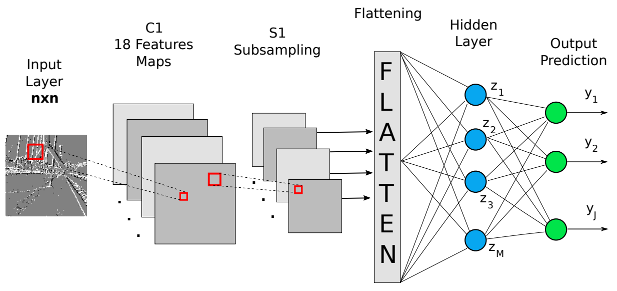

Fig. 3. Three-Layer Spiking Neural Network.

In a second topology, a Hidden Layer with M neurons is added between the flattening module and the classifier/prediction layer, as illustrated in Fig. 3. This time, output neuron state v_j accumulates post-synaptic spikes produced in the new hidden layer, using the corresponding set of synaptic weights $|w_{hj}^{(2)}|$. The superscript (2) indicates that the parameter corresponds to the weights of the second layer of the network, In this architecture, the hidden layer neuron states v_h depend on the flattening module outputs, which are weighted by their corresponding parameter $|w_{ih}^{(1)}|$.

D. Learning Methods

We propose two learning strategies to compute the set of synapses weights.

1) Single-Layer Frame Based: In this strategy, the training is not performed in the spiking domain [11]. Using the spiking simulator MegaSim, an input sequence $\mathbf{s}$ is applied to the spiking neural network generating a list of events at the output of module "Flattening". A histogram, where each bin corresponds to one neuron, is then obtained by counting the number of times a neuron spikes. The histogram is normalized with respect to the maximum value, resulting in a feature vector with values in the interval $[0, 1]$. This vector $\mathbf{x} = (x_1, ...x_N)^T$, referred as "frame", describes the spatio-temporal information within $\mathbf{s}$.

In the case of the two layer topology (see Fig. 2), a logistic regression classifier is trained using stochastic gradient descent [11]. We follow the mini-batch stochastic gradient descent (MSGD) method [12] without bias, to compute the set of synaptic weights $\mathbf{W}_j$ for each class j (or each neuron j in the classifier layer). Given an input vector $\mathbf{x}$, the probability that corresponds to class j is computed by using the *soft-max* function f:

$$p(y = j|\mathbf{x}, \mathbf{W}_j) = f(\sum_{i=1}^{N} w_{ij}^{(1)} x_i) = \frac{e^{\mathbf{W}_j \mathbf{x}}}{Z} \tag{3}$$

where y is the output of the overall system, $\mathbf{W}_j$ is the set of synaptic weights $w_{ij}^{(1)}$ connecting the flattening module output i to neuron j, and $Z = \sum_j e^{\mathbf{W}_j \mathbf{x}}$ is a factor that normalizes the output to the range [0,1].

The predicted class y of the model for input $\mathbf{x}$ is taken as the maximum probability between the output neurons:

$$y = \underset{j=1,...,J}{\text{argmax}}\,(p(y = j|\mathbf{x}, \mathbf{W}_j)) \tag{4}$$

2) Multi-Layer Frame Based: The architecture in Fig. 3 is a feed-forward multi-layer network, with an additional hidden layer of M neurons which are fully connected between the flattening module output and the classifier/prediction layer.

To compute the output probability of neuron j in the hidden layer, M linear combinations of the input $\mathbf{x}$ are computed as:

$$z_h = g(\sum_{i=1}^{N} w_{ih}^{(1)} x_i) \tag{5}$$

where $h = 1, ..., M$, and $g(.)$ is the $tanh$ non-linear activation function.

The output probability of neuron j in the output prediction layer is now computed as:

$$p(y = j|\mathbf{x}) = f(\sum_{h=1}^{M} w_{hj}^{(2)} z_h) \tag{6}$$

The training of the network is performed in two learning stages. First, the multi-layer network is trained as a regular network using the flattening module output "frames" $\mathbf{x}$. The classical back-propagation algorithm is employed to obtain the weight values $w_{ih}^{(1)}$ and $w_{hj}^{(2)}$. Dropout regularization is also used. Then, Megasim loads the training set into the topology that uses the obtained weights which were scaled to an integer value (see Section II-D3).

Then, from the spikes generated by the new hidden layer a new set of "frames" $\mathbf{z}$ are computed. This time, the histograms count the number of times a hidden neuron fires given an input sequence $\mathbf{s}$. With these new $\mathbf{z}$ vectors, we train the weights of the second layer $w_{hj}^{(2)}$ with the same procedure detailed in Section II-D1.

3) From Static to Spiking: After training, the synapse weights have a real value in $\Re$. When mapping to the SNN framework, we scale them to an integer value in $\mathbb{Z}$ proportional to the neuron threshold. In the simulations, this value is about 10e6.

E. DVS Scene Recognition

The SNN system produces a continuous stream of spiking events at the output classification layer. The classifier accumulates the number of times that an output neuron j spikes and predicts with the maximum value, within a window of a fixed number of input DVS events. Fig. 4 shows the recognition accuracy of the different architectures versus the percentage of input events. As can be seen, the best results are obtained when all 100% of events are considered.

III. Results

The system employs the recorded Places-DVS dataset[2], mentioned in Section II-A, to train the SNN. It chooses 1000

[2]http://www2.imse-cnm.csic.es/caviar/SCENES_DVS.html

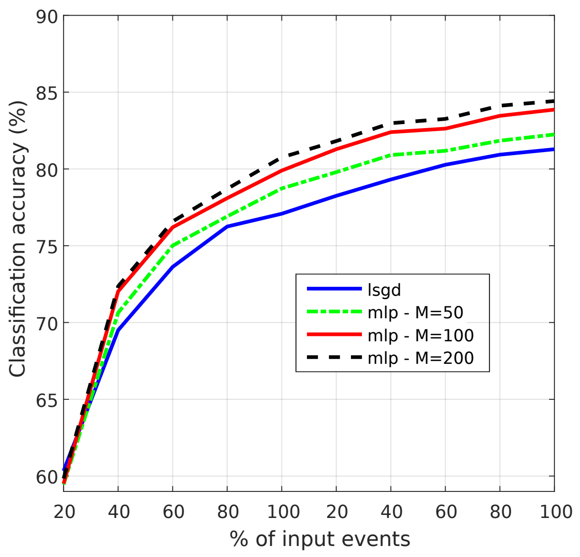

Fig. 4. Classification results for each SNN architecture as a function of the percentage of sequential input events per sample.

TABLE I
CLASSIFICATION RESULTS FOR EACH ARCHITECTURE

Architecture	Classification Accuracy (%)
Single-Layer (lsgd)	81.2 ± 2.8
Multi-Layer: M=50	82.2 ± 3.8
Multi-Layer: M=100	83.8 ± 2.4
Multi-Layer: M=200	$\mathbf{84.4} \pm 3.6$

TABLE II
CONFUSION MATRIX (LSGD) (%)

	bridge	street	stairs	corridor
bridge	**84.4**	12.1	2.5	1.0
street	10.6	**72.9**	11.0	5.5
stairs	5.7	8.8	**75.6**	9.9
corridor	2.0	4.0	8.5	**85.5**

samples from each class randomly. We applied a 5 cross-validation methodology by randomly splitting the total dataset into 5 sets of 200 samples. At each iteration, each set is preserved for the tests, and the remaining 800 samples are employed for learning and validation. Final results, correspond to the average obtained on each test set.

Table I depicts the accuracy of the two architectures, single and multi-layer. The interval from the mean value is obtained from the 5 cross-validation results. For the multi-layer, we test three values of the number of hidden neurons $M = 50, 100, 200$, increasing the complexity of the classifier. As expected, the multi-layer topology with the maximum number of hidden layer neurons, i.e. the more complex, gets the best results. It is not shown here, but further increasing the value of M does not improve the overall results.

Table II shows a confusion matrix obtained using the LSGD classifier. We choose the classifier obtaining the worst result, because it better illustrates the confusion between classes. As can be seen, bridge and corridor are the classes with highest accuracy. They correspond to views with a specific structure, the walls and doors for corridor, or the way and the railing for the bridge. On the other hand, the street class can have several components in the view, i.e. buildings, trees, cars, etc., incrementing their information and giving more problems to the classifiers to generalize the class. The same happens with the stairs class, as can be appreciated on Fig. 1.

IV. CONCLUSIONS

We present two Event-Driven neuromorphic architectures for performing scene context classification on visual data captured by a DVS camera. The results shown confirm the suitability of this approach for performing the proposed cognitive task.

ACKNOWLEDGMENTS

This work was funded by EU H2020 grants 644096 (ECOMODE) and 687299 (NEURAM3), and by Spanish grant from the Ministry of Economy and Competitivity TEC2015-63884-C2-1-P (COGNET) (with support from the European Regional Development Fund).

REFERENCES

[1] M. Szummer and R. W. Picard, "Indoor-outdoor image classification," in *International Workshop on Content-Based Access of Image and Video Database*, Jan 1998, pp. 42–51.
[2] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128*128 120db 15us latency asynchronous temporal contrast vision sensor," *JSSC*, vol. 43, no. 2, pp. 566–576, 2008.
[3] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB Dynamic Range Frame-Free PWM Image Sensor With Lossless Pixel-Level Video Compression and Time-Domain CDS," *IEEE J. of Solid-State Circ.*, vol. 46, no. 1, pp. 259–275, Jan. 2011.
[4] T. Serrano-Gotarredona and B. Linares-Barranco, "A 128x128 1.5sensitivity 0.9sensor using transimpedance preamplifiers," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 3, pp. 827–838, 2013.
[5] M. Guo, J. Huang, and S. Chen, "Live demonstration: A 768×640 pixels 200Meps dynamic vision sensor," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–1.
[6] B. Son, Y. Suh, S. Kim, H. Jung, J. S. Kim, C. Shin, K. Park, K. Lee, J. Park, J. Woo, Y. Roh, H. Lee, Y. Wang, I. Ovsiannikov, and H. Ryu, "4.1 A 640x480 dynamic vision sensor with a 9um pixel and 300Meps address-event representation," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 66–67.
[7] J. Pérez-Carrasco *et al.*, "Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward convnets," *PAMI*, vol. 35, no. 11, pp. 2706–2719, 2013.
[8] I. A. Lungu, F. Corradi, and T. Delbrck, "Live demonstration: Convolutional neural network driven by dynamic vision sensor playing roshambo," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017.
[9] MegaSim. Available on may 2018. [Online]. Available: https://bitbucket.org/bernabelinares/megasim
[10] M. Sivilotti, "Wiring considerations in analog VLSI systems with application to field-programmable networks," *PhD, Computation and Neural Systems, Caltech, Pasadena California*, 1991.
[11] E. Stromatias, M. Soto, T. Serrano-Gotarredona, and B. Linares-Barranco, "An Event-Driven Classifier for Spiking Neural Networks Fed with Synthetic or Dynamic Vision Sensor Data," *Frontiers in Neuroscience*, vol. 11, p. 350, 2017.
[12] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *International Conference on Computational Statistics*. Physica-Verlag HD, 2010, pp. 177–187.