Mapping from Frame-Driven to Frame-Free Event-Driven Vision Systems by Low-Rate Rate-Coding and Coincidence Processing. Application to Feed-Forward ConvNets

J. A. Pérez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen and B. Linares-Barranco

***Abstract* - Event-driven visual sensors have attracted interest from a number of different research communities. They provide visual information in quite a different way from conventional video systems consisting of sequences of still images rendered at a given "frame rate". Event-driven vision sensors take inspiration from biology. Each pixel sends out an event (spike) when it senses something meaningful is happening, without any notion of a frame. A special type of Event-driven sensor is the so called Dynamic-Vision-Sensor (DVS) where each pixel computes relative changes of light, or "temporal contrast". The sensor output consists of a continuous flow of pixel events which represent the moving objects in the scene. Pixel events become available with micro second delays with respect to "reality". These events can be processed "as they flow" by a cascade of event (convolution) processors. As a result, input and output event flows are practically coincident in time, and objects can be recognized as soon as the sensor provides enough meaningful events. In this paper we present a methodology for mapping from a properly trained neural network in a conventional Frame-driven representation, to an Event-driven representation. The method is illustrated by studying Event-driven Convolutional Neural Networks (ConvNet) trained to recognize rotating human silhouettes or high speed poker card symbols. The Event-driven ConvNet is fed with recordings obtained from a real DVS camera. The Event-driven ConvNet is simulated with a dedicated Event-driven simulator, and consists of a number of Event-driven processing modules the characteristics of which are obtained from individually manufactured hardware modules.**

***Indexing Terms:* Feature Extraction, Convolutional Neural Networks, Object Recognition, Spiking Neural Networks, Event Driven Neural Networks, Bio-inspired Vision, High Seed Vision.**

I. INTRODUCTION

In 2006 Delbrück presented the first Event-Driven Dynamic Vision Sensor (DVS) [1]-[2], inspired by Kramer's transient detector concept [3]. This was followed and improved by other researchers [4]-[5]. The DVS presents a revolutionary concept in vision sensing, as it uses an Event-driven Frame-less approach to capture transients in visual scenes.

A DVS contains an array of pixels (i,j) where each pixel senses local light I_{ij} and generates an asynchronous "address event" every time light changes by a given relative amount C > 1 (if light increases: when $I_{ij}(t)/I_{ij}(t_o) = C$, or if light decreases: when $I_{ij}(t)/I_{ij}(t_o) = 1/C$). The "address event" consists of the pixel coordinates (x_{ij},y_{ij}) and sign s_{ij} of the change (increment or decrement). This "flow" of asynchronous events is usually referred to as "Address Event Representation" (AER). Every time a DVS pixel generates such event, the event parameters (x_{ij},y_{ij},s_{ij}) are written on a high speed asynchronous digital bus with nano-second delays. A DVS pixel typically generates one to four events (spikes) when an edge crosses it. DVS output consists of a continuous flow of events (spikes) in time, each with sub-microsecond time resolution, representing the observed moving reality as it changes, without waiting to assemble or scan artificial time-constrained frames (images).

As an illustration, Fig. 1 shows the event flow generated by a DVS when it observes a black 400*Hz* rotating disk with a white dot. On the right, events are represented in 3D coordinates (x,y,t). When a pixel senses a dark-to-bright transition it sends out positive events (dark dots in Fig. 1), and when it senses a bright-to-dark transition it sends out a negative event (gray dots in Fig. 1). Appendix 1 explains the operation of a typical DVS camera in more detail. The flow of events generated by a DVS can be captured with an event logger board [6]-[7], and written on a file with corresponding time-stamps. This file contains a list of signed time-stamped events (t,x,y,s).

Recorded time-stamped events can be processed off-line to perform filtering, noise removal, shape detection, object recognition, and other operations. However, it is more desirable to develop event-driven processing hardware to process events as they are generated by the DVS, without time-stamping them, and to operate in true real time. For example, some event-driven convolution processors have been recently reported for performing large programmable kernel 2D convolutions on event flows [8]-[10]. Appendix 2 briefly explains the operation of a typical AER (Address Event Representation) programmable kernel convolution chip. One very interesting property of this event-driven processing is what we call here "*pseudo-simultaneity*" or "*coincidence*" between input and output event flows. This concept is illustrated with the help of Fig. 2. A vision sensor is observing a flashing symbol that lasts for 1ms. The sensor then sends its output to a 5-layer Convolutional Neural Network for object recognition, as shown in Fig. 2(a). In the case of conventional Frame-Driven sensing and processing, the sequence of processing results would be as depicted in Fig. 2(b). Assuming sensor and each processing stage respond in 1ms, the sensor output image would be available during the next mili second

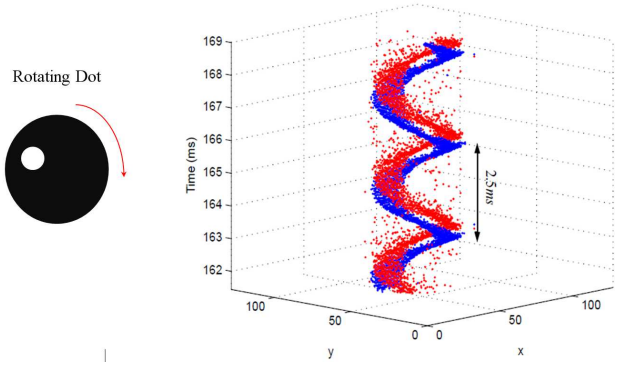


Fig. 1: Example illustration of DVS camera output event flow when observing a black rotating disk with a white dot, rotating at 400*Hz*.

Authors are with the Instituto de Microelectrónica de Sevilla (IMSE-CNM-CSIC) Spain, the Dpto. Teoría de la Señal, ETSIT, University of Sevilla, Spain, and with Nanyang Technological University, Singapore.

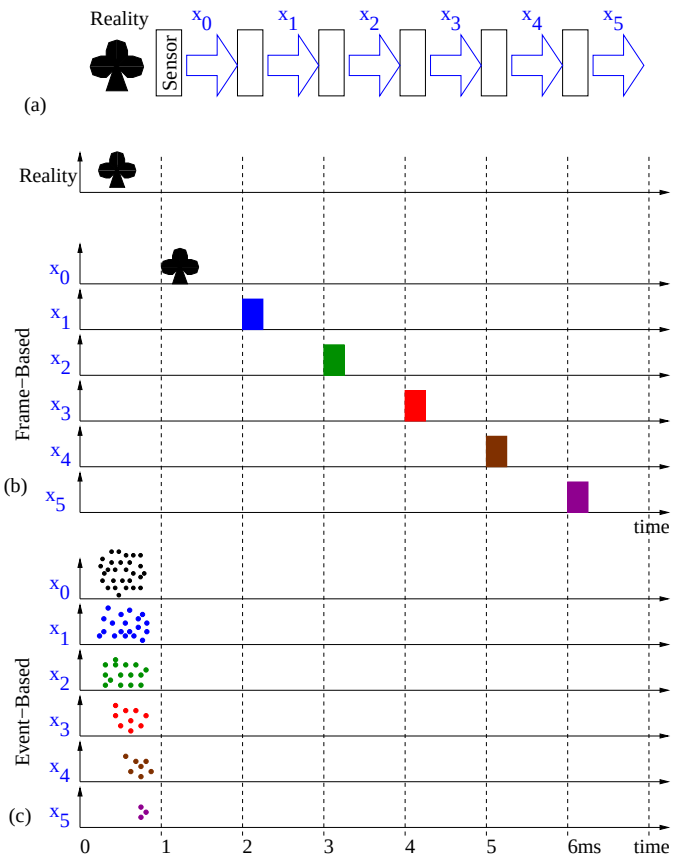


Fig. 2: Illustration of pseudo-simultaneity or coincidence property in a multi layer event-driven processing system. (a) Vision system composed of Vision Sensor and five sequential processing stages, like in a ConvNet. (b) Timing in a Frame-constraint system with 1ms frame time for sensing and per stage processing. (b) Timing in an Event-driven system with micro-second delays for sensor and processor events.

after the flash. Then each sequential stage would provide its output 1ms after receiving its input. Therefore, recognition (last stage output) becomes available 6ms after the symbol flashed. Fig. 2(c) shows the equivalent when sensing and processing with event-driven hardware. Pixels in the sensor create and send out events as soon as they sense a light change, with micro seconds delay [1],[2],[4],[5]. This way, the sensor output events at x_o are in practice simultaneous to the flashing symbol in reality. The first event-driven stage processes events as they flow in, with sub-micro second delays [8]-[11]. As soon as sufficient events are received representing a given feature, output events will be available. Thus, the output feature event flow at x_1 is in practice coincident with the event flow at x_o. The same happens for the next stages. Therefore, recognition at x_5 becomes available during the first mili second, as soon as the sensor provides sufficient events for correct recognition.

This *pseudo-simultaneity* or *coincidence* property becomes very attractive for event-driven processing systems comprising a large number of cascaded event-driven processors with or without feedback, as the overall output can be available as soon as sufficient meaningful input events are provided. This contrasts strongly with state-of-the-art frame-driven vision sensing and processing, where images are first detected by a camera, and then transferred to an image processor.

In this paper we focus on vision systems comprising an event-driven sensor and a large number of event-driven processing modules used to perform object recognition tasks[1]. To do so, we will concentrate on a particular type of bio-inspired vision processing structures called Convolutional Neural Networks (ConvNets) [12]. Reported ConvNets operate based on Frame-Driven principles, and are trained by presenting them with a database of training static images (frames). On the other hand, training of event-driven processing modules is still an open research problem. Some preliminary and highly promising work on this can be found in literature [19]-[20]. However, its application to large scale systems is presently not practical. Therefore, in this paper we present an intermediate solution. First, we build a database of training images (frames) by collecting events from a DVS camera during fixed time intervals. Second, we train a Frame-driven ConvNet with this database to perform object recognition. Third, we map the learned parameters of the Frame-Driven ConvNet to an Event-Driven ConvNet, and finally we fine-tune some extra available timing-related parameters of the Event-Driven ConvNet to optimize recognition. To do this process, we provide a methodology for mapping the properly trained Frame-driven ConvNet into its corresponding Event-driven version. We will then illustrate this with two example ConvNet exercises. One for detecting the angle of rotated DVS recordings of walking human silhouettes, and the other for recognizing the symbols of poker cards when browsing the card deck in about one second in front of a DVS.

The paper is structured as follows. The next Section discusses timing differences between vision in frame-driven and event-driven representations. Section III presents the mapping method from a frame-driven system neuron to an event-driven system neuron. Sections IV and V present two example ConvNet systems that use DVS recordings from real DVS retina chips. In Section IV the example targets a problem where the time constants of the observed world are similar to those we humans are used to, while in the experiment in Section V illustrates the situation for higher speed observed realities where DVS performance is pushed to its limits. Finally Sections VI and VII present some discussions and the conclusions.

II. TIMING IN FRAME-DRIVEN VS. EVENT-DRIVEN VISION REPRESENTATION

In a Frame-driven representation visual processing system "*Reality*" is sensed as binned into time compartments of duration T_{frame}. The implicit assumption is that the time constant $\tau_{reality}$ associated to the changes in "*Reality*" is larger than T_{frame} or, at most, similar. If $\tau_{reality}$ is much larger than T_{frame} (Reality moves slowly) then many subsequent video frames would be quite similar and redundant. An image capturing and processing system working on a frame by frame basis would repeat complex image processing and recognition algorithms over a similar input, wasting computing resources. If $\tau_{reality}$ is much smaller than T_{frame} (Reality moves very fast) then subsequent video frames would be considerably different,

1. As discussed in Appendix 2, with present day technology it is feasible to develop compact hardware with thousands of event-driven convolution modules [28].

making it difficult or impossible to track objects (for example, many flies in a box). Optimally, one would desire to adjust T_{frame} to be close to $\tau_{reality}$ so that subsequent frames are different enough to justify the computing resources employed, but still similar enough to be able to track changes.

In an Event-driven vision sensing and processing system, frames need not be used. For example, in Event-driven temporal contrast retina sensors (DVS), pixels generate output events representing "*Moving Reality*" with time constants that adapt naturally to $\tau_{reality}$. In the particular case of feed-forward multi-layer ConvNets, subsequent layers extract visual features which are simple and short-range in the first layers and progressively become more and more complex and longer-range in subsequent layers until specific full-scale objects are recognized. Typically, first layers extract edges and orientations at different angles and scales, using short-range but dense projection fields (receptive fields). Subsequent layers group these simple features progressively into gradually more sophisticated shapes and figures, using longer range but sparser projection fields. Here we assume that the processing time constants associated with the first feature extraction layers is faster than those associated with later layers. This way early feature extraction layers would be short-range both in space and time, while later feature grouping layers would be longer-range also in both space and time. Note that this makes a lot of sense, since simple features (such as short edges) need to be sensed instantly, while for recognizing a complex shape (like a human silhouette) it would be more efficient to collect simple features during a longer time to be more confident. For example, if we observe a walking human silhouette, at some instants we may not see a leg or an arm, but if we see them at other instants, we know they are there. Consequently, in a Frame-free Event-driven sensing and processing system, we have the extra feature of adapting the time constant of each processing layer independently. This provides extra freedom degrees to optimize overall recognition, which is not directly available in Frame-driven recognition systems.

At present, however, Frame-driven vision machine learning algorithms are much more developed than their Event-driven counterparts. For example, over the last few decades powerful and highly efficient training algorithms have been developed and applied for Frame-driven ConvNets, making them practical and competitive for a variety of real world applications [12]-[18]. Some researchers are presently exploring the possibility of training Event-driven systems, with promising results [19]-[21]. But this field is still under development.

In the next Section we describe a method for mapping the parameters of a properly trained Frame-driven neuron into the equivalent Event-driven Frame-free parameters. We then illustrate this by applying it in ConvNet visual recognition systems that use real recordings from a Frame-free Event-driven DVS retina chip.

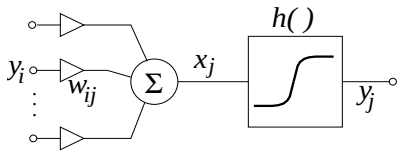


Fig. 3: Conventional individual neuron used in Frame-driven systems, by freezing time during each frame, and resetting its state for each frame.

III. GENERIC MAPPING METHODOLOGY

A. Frame-driven Individual Neuron

Fig. 3 shows the computational diagram of a typical neuron in a Frame-driven representation system. Input signals y_i come from the *i-th* neuron of the receptive field RF_j of neuron *j*, weighted by synaptic weights w_{ij}. Input signals y_i belong to range [0, A_i] or [-A_i, A_i]. Let us call $y = \hat{y}A$, so that $\hat{y}$ is normalized to unity. The state x_j of neuron *j* is reset for each frame, and computed for its receptive field for the present frame as

$$x_j = \sum_{i \in RF_j} y_i w_{ij} = \sum_{i \in RF_j} A_i \hat{y}_i w_{ij} = A_{RF_j} \hat{x}_j$$
$$\hat{x}_j = \sum_{i \in RF_j} \hat{y}_i w_{ij} \tag{1}$$

where we have assumed that all A_i coefficients are the same for all neurons *i* of the RF_j receptive field $A_i = A_{RF_j}$. After this, the neuron state goes through a sigmoidal function $h(\cdot)$, which we may define as[2] [22]

$$y_j = h(x_j) = A_j \tanh(S_j x_j) = A_j \tanh(S_j A_{RF_j} \hat{x}_j)$$
$$\hat{y}_j = \tanh(S_j A_{RF_j} \hat{x}_j) \tag{2}$$

We can describe this using only normalized variables as

$$\hat{y}_j = \hat{h}(\hat{x}_j)$$
$$\hat{x}_j = \sum_{i \in RF_j} \hat{y}_i w_{ij} \tag{3}$$

with $\hat{h}(z) = \tanh(S_j A_{RF_j} z) = (1/A_j) h(z/A_{RF_j})$. A piece-wise-linear approximation of $\hat{h}(\cdot)$ can be defined as

$$h_{pwl}(x) = \begin{cases} x & \text{if} \quad |x| \le 1 \\ x/|x| & \text{if} \quad x \ge 1 \end{cases} \tag{4}$$

Fig. 4 shows the three nonlinear functions $h(x)$, $h_{pwl}(x)$, and $\hat{h}(x)$ for $S = 2/3$ and $A_j = A_{RF_j} = 1.7159$.

B. Event-Driven Individual Neuron

Fig. 5 shows a schematic computational diagram of an Event-driven (spiking signal) neuron. In this case, time plays a crucial role, as opposed to the previous case where time is frozen during all computations corresponding to a frame. Now the neural state x'_j evolves continuously with time. Fig. 5

2. LeCun [22] suggested setting $A = 1.7159$ and $S = 2/3$ to optimize learning speed and convergence in ConvNets.

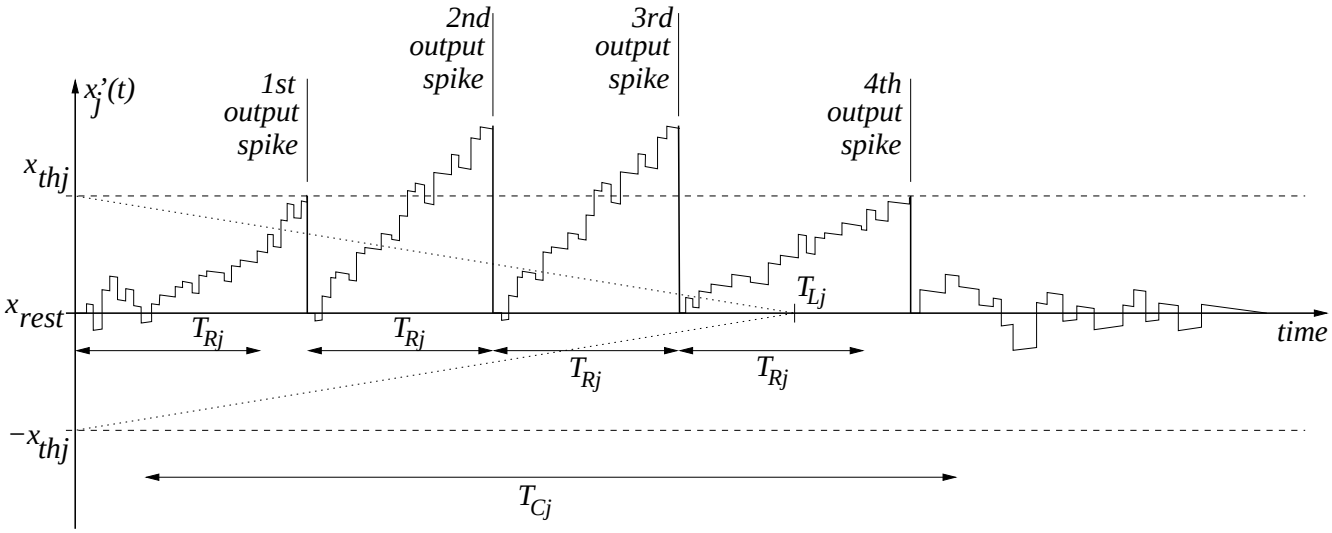


Fig. 6: Illustration of a typical state evolution and spike production sequence for a spiking neuron with leak and refractory period

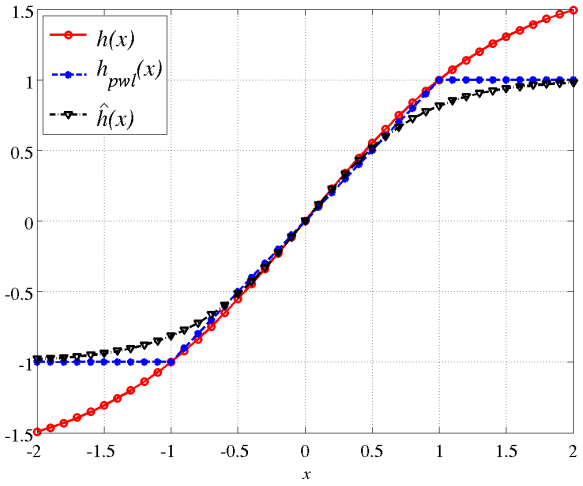


Fig. 4: Comparison between the three nonlinear functions $h(x)$, $h_{pwl}(x)$, and $\hat{h}(x)$

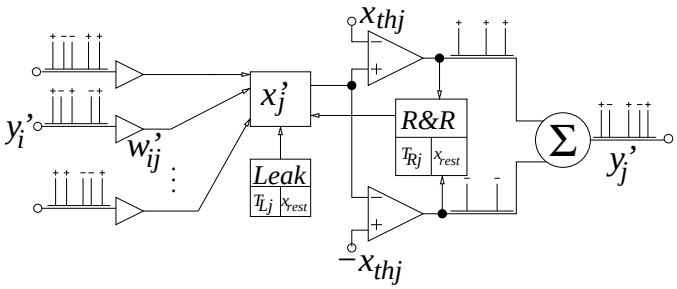


Fig. 5: Computational Block diagram of Event-Driven Neuron

represents state x'_j as being held in a box, while the elements capable of altering it have been drawn with arrows pointing towards this box. These elements are: (a) synaptic connections, (b) leak, and (c) a "reset and refractory" (*R&R*) element.

Pre-synaptic neurons belonging to the receptive field send spikes in time. In general, spikes carry a positive or negative sign, and synaptic weights also have a positive or negative sign. In certain implementations (such as biology) positive and negative events (spikes) are separated into separate paths. Our analyses are not affected by how this is implemented physically. Each pre-synaptic spike will contribute to a certain increment or decrement $\Delta x'$ in the neuron state x'_j proportional to the corresponding synaptic weight $|w'_{ij}|$. The neuron state x'_j will accumulate all these contributions over time, and at a given instant may have a positive or negative accumulated value.

Fig. 6 shows an example of neural state evolution and spike production. Let us define a characteristic time T_{Cj} for neuron *j*. A neuron can be considered a specific feature detector for the collection of spatio-temporal input spikes it receives. For example, neurons in cortex layer V1 spike when they detect sequence of spikes from the retina representing edges at specific scales, orientations, and positions, within a characteristic time interval. A neuron at a higher layer may be specialized in detecting specific shapes, like an eye, nose, etc. Such a neuron would generate spikes when the collection of input spikes from prior neurons represents a collection of edges and shapes that when put together during a characteristic time interval resemble an eye, nose, etc. In Fig. 6 we have represented T_{Cj} as a characteristic time during which neuron *j* receives a meaningful collection of spikes (representing the specific feature of neuron *j*) that produce a systematic increase in its state. Every time the state x'_j reaches one of the thresholds $\pm x_{thj}$, the *R&R* element will reset the state to its resting level x_{rest}, while guaranteeing also a minimum separation between consecutive spikes T_{Rj}, called the "Refractory Time" of this neuron. This refractory effect is equivalent to the saturation function $h(\cdot)$ in the frame-driven system, as it limits the maximum output spike event rate.

If all neurons *i* of the receptive field of neuron *j* have the same characteristic time T_{Ci} and/or refractory time T_{Ri}, we can define the "characteristic time gain" of neuron *j* as

$$g_{\tau j} = T_{Cj}/T_{Ci} \tag{5}$$

and the "refractory time gain" of neuron *j* as

$$g_{Rj} = T_{Rj}/T_{Ri} \tag{6}$$

We will use these definitions later.

Neurons will not accumulate all historic incoming spikes contributions (similarly, in the frame-driven case, neurons ignore information from previous frames). Since a neuron is

interested in grouping lower level features from previous neurons during a characteristic time T_{Cj}, its state x'_j is subject to a continuous leak that will drive its value towards x_{rest} with a characteristic leak time constant. Fig. 6 shows a linear leak for the neuron state, with a leak rate of value $LR_j = |x_{thj}/T_{Lj}|$.

C. Signal Encoding in Frame-Free Event-Driven Systems. Low-Rate Rate-Coding or Coincidence Processing

In traditional frame-driven neural computing systems, neuron states and neuron output values are usually represented with floating point precision. In some specialized accelerated hardware implementations, 8-bit signed integer representation is used [23]. Still, this representation presents a high dynamic range, since the ratio between the full range and the smallest step is $2^8 = 256$. Note that in a recognition system, the output of a neuron is not required to present such a high dynamic range, since it only has to signal whether a feature is present or not, or at the most provide a relative confidence which could be provided with coarse steps. For example, in a face detection application, we would not expect it to be critical whether the neurons detecting the nose can use just five values [0, 0.25, 0.50, 0.75, 1.0] to give their confidence, or can use 256 steps in the range [0, 1]. A higher dynamic range might be necessary to represent the visual input. Commercial video, photography and computer screens normally use 8-bit to represent luminance. However, we will assume that our Event-driven visual sensors include a preprocessing step (such as spatial or temporal contrast) that significantly reduces the dynamic range of the signals provided by their pixels. For example, a pixel in the temporal contrast DVS retina we have used, normally provides between 1 to 4 spikes when an edge crosses it.

In the following mathematical developments for mapping from the Frame-driven domain to the Event-driven domain, we will consider that an intensity value in the former is mapped to a spike rate in the latter. Obviously, rate-coding is highly inefficient for high dynamic ranges such as 8-bit, because a neuron would need to transmit 256 spikes to represent a maximally meaningful signal. Although the following mathematical developments have no restrictions in terms of dynamic range, we will always keep in mind that we will in practice apply it to low dynamic range signals. We call this "*Low-Rate Rate-Coding*", and the maximum number of spikes a neuron will transmit during its characteristic time constant will be kept relatively low (for example, just a few spikes, or even as low as one single spike). This maximum number of spikes is T_{Cj}/T_{Rj}. Time T_{Rj} is the minimum inter-spike time needed to signal the presence of a feature, while T_{Cj} is the characteristic time during which this feature might be present during a transient. Thus, let us call "persistency" p_j of neuron j the maximum number of spikes that can be generated by a transient feature during time T_{Cj}

$$p_j = T_{Cj}/T_{Rj} \tag{7}$$

Using all the above concepts and definitions, let us now proceed to mathematically analyze the event-driven neuron and propose a mapping formulation between frame-driven and event-driven neuron parameters.

D. Mathematical Analysis of Event-Driven Neurons

With reference to Fig. 6, let us consider the situation where neuron j becomes active as it receives a collection of properly correlated spatio-temporal input spikes during a time T_{Cj}. These represent the feature to which neuron j is sensitive. In this case, the collection of spikes will produce a systematic increase in neuron j's activity x'_j during time T_{Cj}, resulting to the generation of some output spikes, as illustrated in Fig. 6. If the output spikes are produced with inter-spike intervals larger than the refractory period T_{Rj}, then the number of spikes n_j produced by neuron j during time T_{Cj} satisfies[3]

$$\frac{n_j}{T_{Cj}} = \frac{\left(\sum_{i \in RF_j} n_i w_{ij}'\right) - \Delta x_{Lj}}{x_{thj} T_{Cj}} \tag{8}$$

where Δx_{Lj} is the loss of neural activity x'_j due to leak. We may safely assume that, during a systematic neural activity build up that produces output events, leak does not drive the activity down to the resting level x_{rest}, nor produces a change of its sign. Under this assumption,

$$LR_j = \frac{|\Delta x_{Lj}|}{T_{Cj}} = \frac{x_{thj}}{T_{Lj}} \tag{9}$$

If the systematic activity build up is sufficiently fast, then the neuron activates its refractory mode and will not allow inter-spike intervals shorter than T_{Rj}, or equivalently

$$\frac{n_j}{T_{Cj}} \le \frac{1}{T_{Rj}} \tag{10}$$

as is illustrated in Fig. 6 for the second and third spikes produced by neuron j during time T_{Cj}. To take this into account, the right hand side of eq. (8) needs to saturate to $1/T_{Rj}$. This can be expressed as

$$\frac{n_j}{T_{Cj}} = \frac{1}{T_{Rj}} h_{pwl}\left(\frac{\left(\sum_{i \in RF_j} n_i w_{ij}'\right) - \Delta x_{Lj}}{x_{thj}} \frac{T_{Rj}}{T_{Cj}}\right) \le \frac{1}{T_{Rj}} \tag{11}$$

where $h_{pwl}(\cdot)$ saturates to '1' and is as defined in eq. (4). Using eqs. (7) and (9), eq. (11) becomes

$$\frac{n_j}{p_j} = h_{pwl}\left(\left(\sum_{i \in RF_j} \frac{n_i}{p_j} \frac{w_{ij}'}{x_{thj}}\right) - \beta_j\right) \tag{12}$$

where $\beta_j = T_{Rj}/T_{Lj}$. Noting that β_j will usually tend to be much smaller than unity and that $n_j/p_j \in [-1, 1]$, we can establish a parallelism between bottom eq. (2) and eq. (12) by means of the following mapping

3. Here we are assuming a positive increase in the state ($x'_j > x_{rest}$), reaching the positive threshold and producing positive output events. The analysis is equivalent for the generation of negative events.

Table 1. Summary of Event-Driven Neuron Parameters

x_{thj}	threshold
T_{Cj}	characteristic time
T_{Rj}	refractory time
LR_j	leak rate
$p_j = T_{Cj}/T_{Rj}$	persistency
$g_{\tau j} = T_{Cj}/T_{Ci}$	characteristic time gain
$g_{Rj} = T_{Rj}/T_{Ri} = r_{ij}$	refractory time gain
$\beta_j = T_{Rj}/T_{Lj}$	refractory-leak ratio

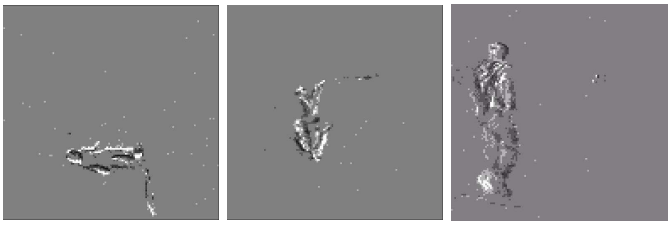


Fig. 7: Example snapshot images obtained by histogramming events during 80ms and rotating the (*x*,*y*) addresses.

$$\hat{y}_j \leftrightarrow \frac{n_j}{p_j} \quad , \quad \hat{y}_i \leftrightarrow \frac{n_i}{p_i}$$

$$w_{ij} \leftrightarrow \frac{w_{ij}'}{x_{thj}}\frac{p_i}{p_j} = \frac{w_{ij}'}{x_{thj}}\frac{g_{Rj}}{g_{\tau j}} \qquad (13)$$

$$\hat{h}(\cdot) \leftrightarrow h_{pwl}(\cdot)$$

Note that the kernel w_{ij}' weights used for the event-driven realization are simply scaled versions of those trained in the frame-based version w_{ij} . Table 1 summarizes the different event-driven neuron parameters discussed. As we will see in the rest of the paper, when applying this mapping to ConvNets, we will use the same mapping for all neurons in the same ConvNet layer.

It is interesting to highlight that in a Frame-driven system, neuron states $\hat{x}_j$ can be interpreted as showing how much they have changed during the frame time T_{frame} after being reset, and this frame time can in turn be interpreted as the "characteristic time" T_C of all neurons in the system. When mapping from a frame-driven description to an event-driven one, all neurons could therefore be made to have identical timing characteristics. However, as we will see later on, neurons in different ConvNet layers will be allowed to have different timing characteristics to optimize recognition performance and speed.

IV. EVENT-DRIVEN CONVNET FOR HUMAN SILHOUETTE ORIENTATION RECOGNITION

As an illustrative example of scenes moving at speeds we humans are used to, we trained a frame-driven version of a ConvNet to detect the orientation of individual human walking silhouettes. We used a 128x128 pixel DVS (dynamic vision sensor) camera [5] to record event sequences when observing individual people walking. Fig. 7 shows some (*x*,*y*) rotated sample images obtained by collecting DVS recorded events during[4] 80*ms*. White pixels represent positive events (light changed from dark to bright during these 80*ms*), while black pixels represent negative events (light changed from bright to dark). One person walking generates about 10-20*keps* (kilo events per second) with this DVS camera. From these recordings, we generated a set of images by collecting events during frame times of 30*ms*. From these reconstructed images, we randomly assigned 80% for training and 20% for testing learning performance. Each 128x128 pixel reconstructed image was downsampled to 32x32 and rotated 0º, 90º, 180º or 270º.

Table 2. ConvNet Structure

	C1	S2	C3	S4	C5	C6
Feature Maps (FM)	6	6	4	4	8	4
FM size	28x28	14x14	10x10	5x5	1x1	1x1
kernels size	10x10	-	5x5	-	5x5	1x1
kernels	6	-	24	-	32	32
weights	600	-	600	-	800	32
trainable weights	0	-	600	-	800	32

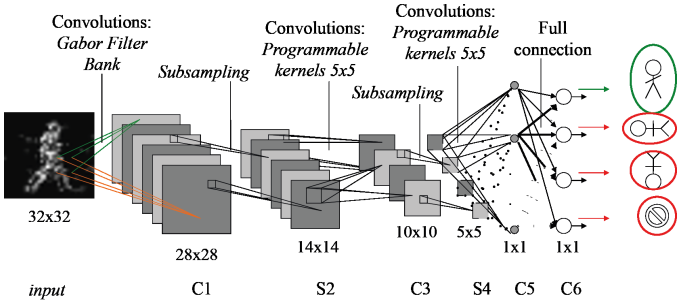


Fig. 8: ConvNet structure for human silhouette orientation detection

The training set images were used to train the Frame-driven 6 layer Feed Forward ConvNet shown in Fig. 8. Table 2 summarizes the number of Feature Maps (FM) per layer, FM size, kernels size, total number of kernels per layer, total weights per layer, and how many weights are trainable. The first layer C1 performs Gabor filtering at 3 orientations and 2 scales, and its weights are not trained. S layers perform subsampling and subsequent S layers perform feature extraction and grouping. The last layer (C6) is not a feature extraction layer, but a feature grouping layer. It performs a simple linear combination of the outputs of the previous layer. The top three neurons in layer C6 recognize a human silhouette rotated 0º, +/-90º, or 180º. The bottom neuron (noise) is activated when the system does not recognize a human silhouette. Each output neuron fires both positive and negative events, depending on whether it is certain the desired pattern is present or it is certain it is not present.

The weights from the Frame-driven system were then mapped to an event-driven version by using the transformations in eqs. (13). Note that in a Feed forward ConvNet, all neurons in the same layer operate with identical spatial scales (kernel sizes and pixel space sizes). Here, we will

4. Recordings were made by connecting the DVS camera via USB to a laptop running jAER [25]. jAER is an open software for managing AER chips and boards, recording events, playing them back, and performing a variety of processing operations on them. Appendix 3 gives a brief overview of jAER.

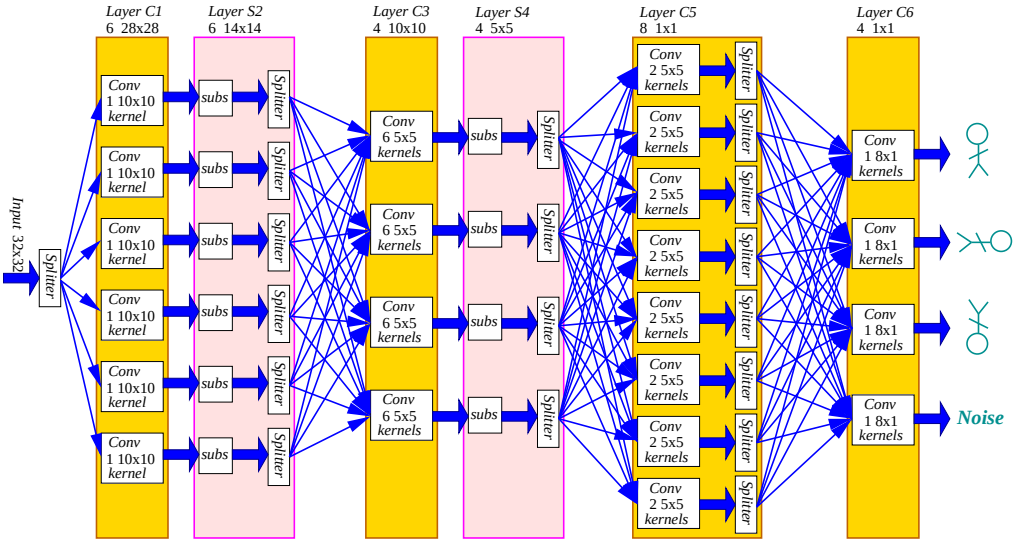


Fig. 9: Schematic block diagram used in simulator AERST

enforce that neurons in the same layer operate with identical temporal scales too, so that each layer extracts spatio-temporal features of the same scales. In Table 1 we would therefore replace index *j* with the layer number index *n*, and index *i* with the previous layer index *n*-1. We also chose $w_{ij} = w_{ij}'$ in eqs. (13), which enforces $g_{Rn} = x_{thn} g_{\tau n}$.

However, eqs. (5)-(7),(13) offer a high degree of freedom to map parameters. We first followed the heuristic rationale outlined below, but afterwards we ran simulated annealing optimization routines to adjust the different parameters for optimum performance.

The temporal patterns generated by the 128x128 DVS camera when observing walking humans are such that a minimum time of about 10-20*ms* (about 100-600 events) is needed to reconstruct a human-like silhouette[5]. We therefore set the "refractory time" of the last refractory layer (C5) as $T_{R5} \approx 10ms$. On the other hand, the persistency of a moving silhouette is on the order of 100*ms* (collecting events for over 100*ms* fuzzyfies the silhouette). Thus $T_{C5} \approx 100ms$. For layer C1, short range edges can be observed with events separated about 0.1*ms* in time. We therefore set $T_{R1} \approx 0{,}1ms$. For layer C3 we chose an intermediate value $T_{R3} \approx 0{,}5ms$. For the thresholds, we picked a value approximately equal to twice the maximum kernel weight projecting to each layer. For the leak rates we picked an approximate ratio of 2:1 between consecutive layers, so that the last layer C6 has a leak rate of $1s^{-1}$. With these criteria, and considering that $g_{Rn} = x_{thn} g_{\tau n}$, the resulting list of parameters describing the event-driven system are shown in Table 3.

Despite this heuristic method of obtaining a set of timing parameters for the 6 layer event-driven ConvNet, we also used optimization routines to optimize these parameters for best recognition rate, as mentioned in the next Section and described in detail in Appendix 5.

Table 3: Parameters adjusted by heuristic rationale

layer	T_{Rn}	x_{thn}	*leak rate*	T_{Ln}	T_{Cn}	p_n	g_{Rn}	$g_{\tau n}$	β
C1	0.1*ms*	0.6	$10s^{-1}$	60*ms*	5*ms*	40	-	-	0.0017
C3	0.5*ms*	1	$5s^{-1}$	200*ms*	25*ms*	40	5	5	0.005
C5	10*ms*	5	$2s^{-1}$	2.5*s*	100*ms*	10	20	4	0.004
C6	-	1.5	$1s^{-1}$	1.5*s*	-	-	-	-	-

A. Results

For testing the event-driven system we used new recordings from the 128x128 pixel DVS camera observing people. These recordings where down-sampled to the 32x32 input space.

To run the simulations on these recordings we used the Address-Event-Representation event-driven simulator AERST (AER Simulation Tool) [24]. This simulator is briefly described in Appendix 4. It uses AER processing blocks, each with one or more AER inputs and one or more AER outputs. AER outputs and inputs are connected through AER point-to-point links. Therefore, the simulator can describe any AER event-driven system through a netlist of AER blocks with point-to-point AER links. A list of DVS recordings is provided as stimulus. The simulator looks at all AER links and processes the earliest unprocessed event. When an AER block processes an input event, it may generate a new output event. If it does, it will write a new unprocessed event on its output AER link. The simulator continues until there are no unprocessed events left. At this point, each AER link in the netlist contains a list of time-stamped events. Each list represents the visual flow of spatio-temporal features represented by that link. The resulting visual event flow at each link can be seen with the jAER viewer. Fig. 9 shows the netlist block diagram of the Event-Driven ConvNet system simulated with AERST. It contains 19 splitter modules, 20 AER convolution modules, and 10 subsampling modules. In AERST the user can describe modules by defining the operations to be performed for each

5. A retina with higher spatial resolution (256x256 or 512x512) multiplies the number of events produced (by 4 and by 16, respectively) for the same stimulus, but the events would still be produced during the same 10-20*ms* time interval. Therefore, we conjecture that increasing spatial resolution reduces recognition time, because the 100-600 events for first recognition would be available earlier.

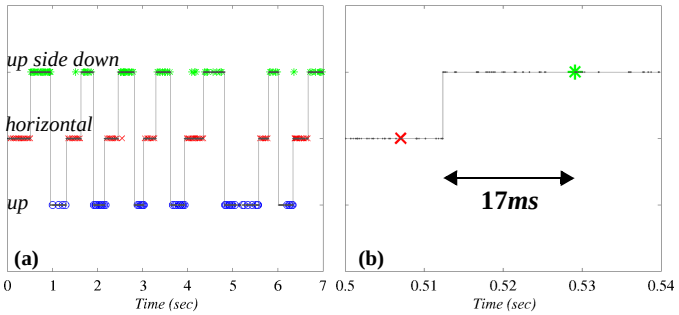


Fig. 10: Recognition performance of the event-driven ConvNet in Fig. 8. Small black dots correspond to input events, circles are output events for 'upright' orientation (0º), crosses for 'horizontal' orientation (90º, 270º), and stars for 'upside down' (180º). (a) 7 sec recording of 20 consecutive orientations, (b) zoom out of 40ms showing a recognition delay of 17ms.

incoming event and include non-ideal effects such as characteristic delays, noise and jitter, limited precision, etc. We described the modules using the performance characteristics of already manufactured AER hardware modules; specifically, available splitter and mapper modules implemented on FPGA boards [6],[9], and dedicated AER convolution chips with programmable kernels [8],[10]. A splitter module replicates each input event received at each of its output ports with a delay on the order of 20-100*ns*. Subsampling modules are implemented using mappers. They transform event coordinates. In this particular case, the mappers are programmed to replace each input event coordinate (*x*,*y*) with $(\lfloor x/2 \rfloor, \lfloor y/2 \rfloor)$, where operand $\lfloor z \rfloor$ is "round to the lower integer". This way, all events coming from a 2x2 square of pixels are remapped to a single pixel. Convolution modules describe AER convolution chip operations with programmable kernels. Convolutions are computed and updated event by event, as described in Appendix 2.

Using the parameters in Table 3 we tested the performance of the Event-driven ConvNet in Fig. 9 when fed with event streams of DVS captured walking human silhouettes rotated 0º, 90º, 180º, and 270º. Each input stream segment consists of 2k events, and their (*x*,*y*) coordinates are consecutively rotated 0º, 90º or 270º, and 180º. Fig. 10 shows input events as small black dots. Output events are marked either with circles ("*upright*" or 0º), crosses ("*horizontal*", or 90º and 270º), or stars ("*upside down*" or 180º). Fig. 10(b) shows a zoom out for the first transition from horizontal to upside down, where the recognition delay is 17*ms*. Average recognition delay was 41*ms*, and the fastest was 8.41*ms*. The overall recognition success rate *SR* for this stream was 96.5%, computed as the average of the success rate per category SR_i. For each category *i*, its success rate is computed as

$$SR_i = \frac{1}{2}\left(\frac{p_i^+}{p_i^+ + p_i^-} + \frac{p_{j\neq i}^-}{p_{j\neq i}^+ + p_{j\neq i}^-}\right) \qquad (14)$$

where p_i^+ (p_i^-) is the number positive (negative) output events for category *i* when input stimulus corresponds to category *i*, and $p_{j\neq i}^+$ ($p_{j\neq i}^-$) are the positive (negative) output events for the other categories. In a perfect recognition situation $p_i^- = 0$ and $p_{j\neq i}^+ = 0$.

Frame Time	Total non-empty Frames	Success Rate
125ms	45	95.6%
100ms	63	96.8%
75ms	84	97.6%
50ms	133	97.7%
30ms	225	96.4%
20ms	339	96.2%
10ms	674	93.5%

Table 4. Performance of Frame-Driven Realization of the Human Silhouette Orientation Detection ConvNet

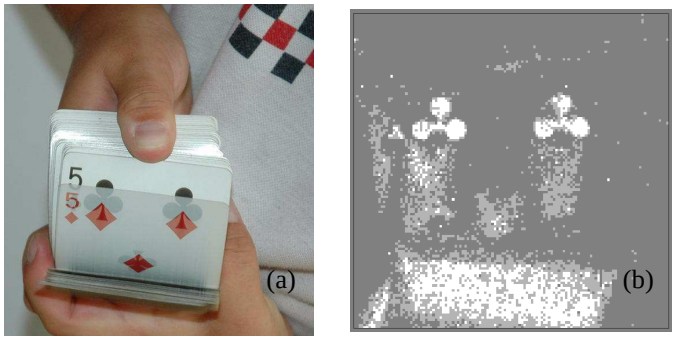


Fig. 11: Fast browsing of a Poker Card Deck. (a) Picture taken with a Frame-Driven Camera. (b) jAER image obtained by collecting events during 5ms.

The parameters in Table 3 were assigned by intuition. Appendix 5 describes the results obtained when using a simulated annealing optimization procedure to optimize these parameters. The resulting optimized parameters are not too different from the ones obtained by intuition, and the recognition rate varied within the range 97.28% to 99.61%.

In order to compare recognition performance with that of a frame-driven ConvNet realization, we used the same sequence of events and built sequences of frames using different frame reconstruction times. Each frame was fed to the frame-driven ConvNet. Table 4 shows, for each frame reconstruction time, the total number of non-empty frames (some frames were empty because the sensor was silent during these times), and the percent of correctly classified frames. As can be seen, the success rate changes with frame reconstruction time, and seems to have an optimum in the range 50-75ms frame time.

V. EVENT-DRIVEN CONVNET FOR POKER CARD SYMBOL RECOGNITION

In this Section we illustrate the method with a second example, more oriented towards high speed sensing and recognition. Fig. 11(a) shows an individual browsing a poker card deck. A card deck can be fully browsed in less than one second. When recording such a scene with a DVS and playing the event flow back with jAER one can freely adjust the frame reconstruction time and frame play back speed to observe the scene at very low speed. Fig. 11(b) illustrates a reconstructed frame when setting frame time to 5ms. The DVS had 128x128 pixels. A poker symbol fits well into a 32x32 pixel patch. We made several high speed browsing recordings, built frames of

Frame Time	Total non-empty Frames	Success Rate
10ms	30	63.3%
7ms	45	77.8%
5ms	54	85.2%
4ms	62	91.9%
3ms	75	94.7%
2ms	84	95.2%
1ms	95	92.6%

Table 5. Performance of Frame-Driven Realization of the Poker Card Symbol Recognition ConvNet

2ms time, and cropped many versions of poker symbols of size 32x32 pixels. We selected a set of best looking frames to build a database of training and test patterns. The topological structure of the Frame-driven ConvNet used for recognizing card symbols was identical to the one described in the previous Section.

The trained Frame-driven ConvNet was mapped to an Event-Driven version, by using the same sets of learned kernel weights and adjusting the timing parameters to the higher speed situation. To provide a proper 32x32 pixel input scene we used an event-driven clustering-tracking algorithm [26] to track card symbols from the original 128x128 DVS recordings, since the instant they appeared until they disappeared. Such time interval ranged typically from about 10-30ms per symbol and the 32x32 crop could contain on the order of 3k to 6k events. We sequenced several of these tracking crops containing all symbols and used the sequences as inputs to the Event-driven ConvNet. We then tested the Event-driven ConvNet with the Event-driven AER simulator used in Section IV and used the same Matlab simulated annealing routines to optimize timing parameters. Table 2 in Appendix 5 shows the optimized parameters and performance results of several optimizations after running simulated annealing. The recognition success rate, measured by eq. (14), varied between 90.1% and 91.6%.

In order to compare the recognition performance with that of a Frame-driven ConvNet realization, we used the same input event recordings to generate frames with different frame times, from 1*ms* to 10*ms*. Then we exposed the originally trained Frame-driven ConvNet to these new frames and obtained the recognition rates shown in Table 5. As can be seen, there is an optimum frame time between 2 and 3*ms*.

Fig. 12 shows an example situation of the output recognition events of the event-driven ConvNet while a sequence of 20 symbol sweeps was presented. The continuous line indicates which symbol was being swept at the input, and the output events are represented by different markers for each category: circles for “club” symbol, crosses for “diamond” symbol, inverted triangles for “heart” symbols, and non inverted triangles for “spade” symbols. Fig. 13 shows the details of the 14ms sequence of the 4th “heart” symbol input in Fig. 12. Fig. 13 contains 14 columns. Each corresponds to building frames using the events that appeared during one mili second at some of the ConvNet nodes. Background gray color

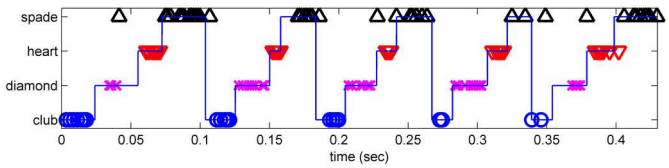


Fig. 12: Recognition performance of the Poker Card Symbol Recognition ConvNet.

represents zero events during this mili second, brighter gray level means a net positive number of events per pixel, while darker gray level means a net negative number of events per pixel. The numbers on the top left of each reconstructed frame indicate the total number of events present for all pixels in that Feature Map during that mili second. Each row in Fig. 13 corresponds to one ConvNet node or Feature Map. The top row corresponds to the 32x32 pixel input crop from the DVS retina. The next 6 rows correspond to the 6 subsampled Feature Maps of the first convolution layer, namely layer S2 14x14 pixel outputs. The next four rows correspond to the 5x5 Feature Map outputs of layer S4. The next 8 rows show the single pixel outputs of layer C5, and the last four rows correspond to the four output pixels of layer 6, each indicating one of the four poker symbol categories. We can see that at the output layer C6 nodes there is a sustained activity for the third row (which corresponds to category “heart”) between mili seconds 6 and 12, which is when the input symbol appeared more clearly. during these 6ms there were 4 positive output events for this category, which we artificially have binned into 1ms slots.

To better illustrate the timing capabilities of multi-layer event-driven processing, we selected a 1ms cut of the input stimulus sequence in Fig. 13. We ran again the simulation for this 1ms input flash and obtained the results shown in Fig. 14. There are five time diagrams in Fig. 14. The top diagram represents a (*y,time*) projection of the DVS retina events. Positive events are represented by circles and negative events by crosses. On top of each diagram we indicate the total number of events in the diagram. The second diagram corresponds to the events in Feature Map 2 of Layer S2. The next diagram represents the events for Feature Map 3 of Layer S4. The next diagram shows the events of all 8 neurons in Layer C5, and the bottom diagram shows the events of all neurons in the output Layer C6. We can see that the neuron of category “heart” in Layer C6 provided one positive output event. Fig. 14 illustrates nicely the “*pseudo-simultaneity*” property or coincidence processing of event-driven multi-layer systems. As soon as one layer provides enough events representing a given feature, the next layer Feature Map tuned to this feature fires events. This property is kept from layer to layer, so that output recognition can be achieved while the input burst is still happening.

In order to characterize the internal representations and dynamics of this event driven network, we show in Appendix 7 reverse correlation reconstructions for the last layers with down to 0.1ms time windows.

VI. DISCUSSION

Event-driven sensing and processing can be highly efficient computationally. As can be seen from the previous

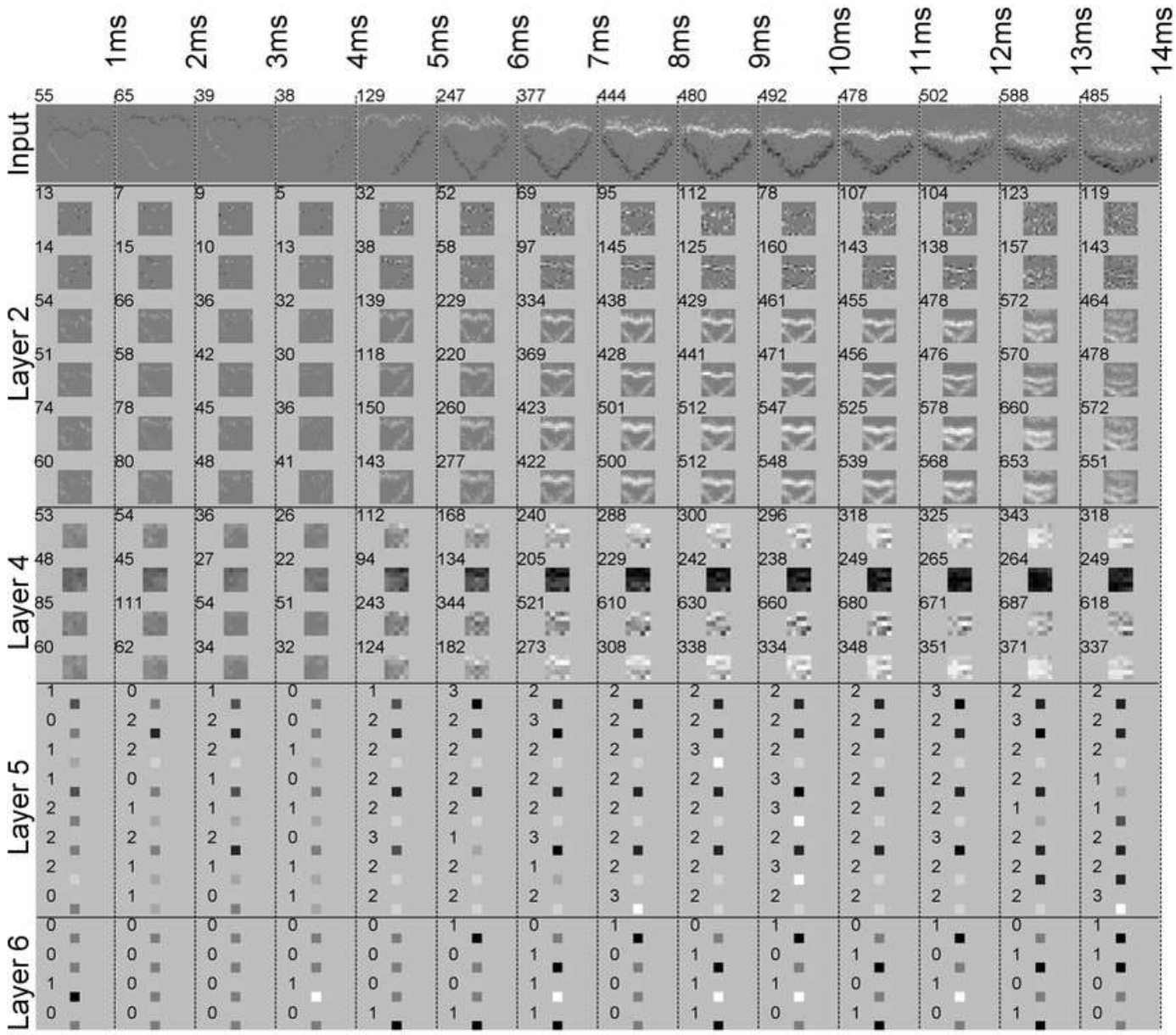


Fig. 13: Image reconstruction for event flows at different ConvNet stages during a 14ms heart symbol card browsing. Images in this figure are built by collecting events during 1ms. Event flows are continuous throughout the full 14ms shown, but are artificially grouped in 1ms bins.Time advances from left to right in steps of 1ms. Images in the same column use events collected during the same ms. Horizontal rows correspond to one single feature map output. Top row is a 32x32 pixel crop of the DVS output tracking a 14ms heart symbol sweep through the screen. Next 6 rows correspond to Layer S2 subsampled 14x14 pixel Feature Maps. Next 4 rows correspond to Layer S4 subsampled 5x5 pixel Feature Maps. Next 8 rows correspond to Layer C5 single pixel features. Bottom 4 rows correspond to the Layer 6 outputs corresponding to the four recognition symbol.

results (for example, Fig. 14), recognition occurs while the sensor is providing events. This contrasts strongly with the conventional Frame-driven approach, where the sensor first needs to detect and transmit one image. In commercial video, frame rate is T_{frame} = 30-40*ms*. Assuming instantaneous image transmission (from sensor to processor) and instantaneous processing and recognition, the output would therefore be available after T_{frame} of sensor reset (the sensor is reset after sending out a full image). In practice, real time image processors are those capable of delivering an output at frame rate; that is, after a time T_{frame} of the sensor making an image available, or, equivalently, after 2xT_{frame} of sensor reset.

Another observation is that in a Frame-driven multi-convolution system like the one shown in Fig. 9, the operations in layer *n* cannot start until operations in layer *n*-1 have concluded. If the system includes feedback loops, then the computations have to be iterated several cycles until convergence, for each frame. However, in the Event-driven approach this is not the case. A DVS camera (or any other Event-driven sensor) produces output events while "*reality*" is actually moving (with micro seconds delay per event). These events are then processed event by event (with delays of around 100*ns*). This effectively makes input and output event flows simultaneous (we have called this *pseudo-simultaneity* or *coincidence* property throughout the paper), not only between the input and output of a single event processor but for the full cascade of processors, as in Fig. 9. Furthermore, if the system includes feedback loops, this coincidence property is retained. Recognition delay is therefore not determined by the number of layers and processing modules per layer, but by the statistical distribution of meaningful input events generated by

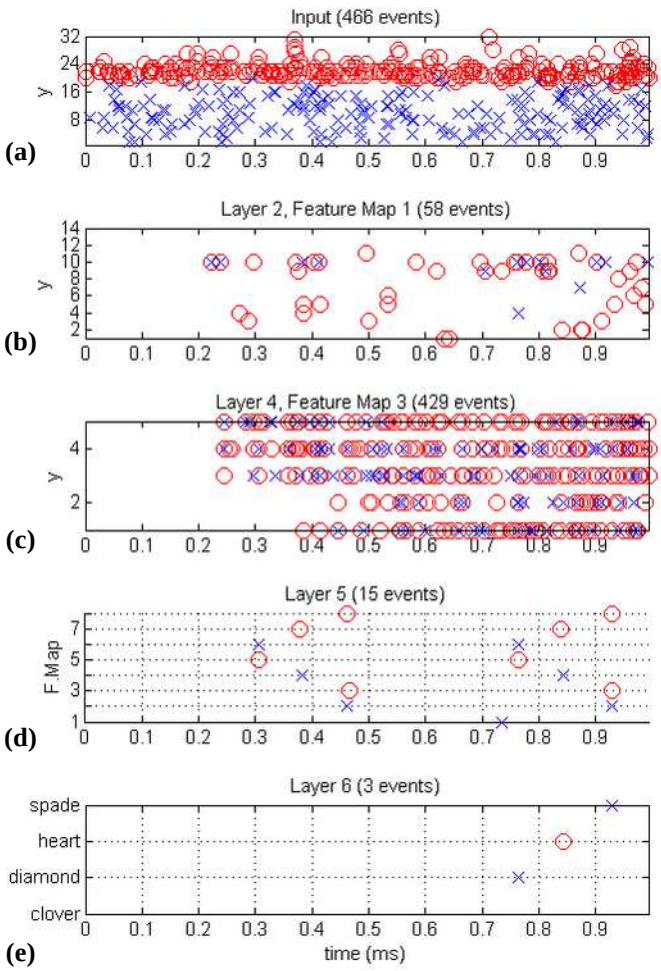


Fig. 14: Events vs time for a simulation that uses as stimulus a 1ms "heart" symbol cut of that in Fig. 13. Positive events are drawn with circles, while negative events use crosses. (a) 1ms event flash from DVS 32x32 crop. (b) Events at 2nd Feature Map FM2 of Layer S2. (c) Events at FM3 of Layer S4. (d) Events at all 8 single pixel FMs of Layer C5. (e) Events at all 4 outputs of Layer C6. Layer C6 produces only 3 events during this 1ms "heart" flash. From these 3 events, only one is positive corresponding to the correct "heart" category.

the sensor. Improving the sensor event generation mechanism would thus in principle improve the overall recognition performance and speed of the full system (as long as it does not saturate). For example, improving the contrast sensitivity of a DVS would increase the number of events generated by the pixels for the same stimulus. Also, increasing spatial resolution would multiply the number of pixels producing output events. This way, more events would be generated during the same time, and the "shapes critical for recognition" would become available earlier.

System complexity is increased in a ConvNet by adding more modules and layers. This makes it possible to increase both the "shape dictionary" at intermediate layers and the "object dictionary" at the output layers. However, in an Event-driven system, increasing the number of modules per layer would not degrade speed response, as long as it does not saturate the communication bandwidth of the inter-module links.

This is one issue to be careful with in Event-driven systems. Event traffic saturation is determined by the communication and processing bandwidth of the different AER links and modules, respectively. Each channel in Fig. 9 has a limited maximum event rate communication bandwidth. Similarly, each module also has a maximum event processing rate. Module (filter) parameters therefore have to be set in such a way that maximum communication and processing event rate is not reached. Normally, the event rate is higher for the first stages (sensor and C1), and decreases significantly for later stages. At least this is the case for the feed forward ConvNets we have studied.

One very attractive feature of event-driven hardware is its ease of scalability. Increasing hardware complexity means connecting more modules. For example, with present day high end ASIC technology it is feasible to place several large size (64x64 or 128x128) ConvModules (about 10) on a single chip, together with companion routers (to program connectivity) and mappers. An array of 10x10 of these chips can be put on a single PCB, hosting on the order of 1k ConvModules. And then, many of these PCBs could be assembled hierarchically. Several research groups are pursuing this type of hardware assembly goals [27],[28],[30],[45]. In Appendix 6 we compare in more detail frame vs. event-driven approaches focusing on hardware aspects.

Regarding the sensor, we have focused our discussions on the DVS camera. However, there are many reported Event-driven sensors for vision and audition. Just to mention a few, there are also plain luminance sensors [31], time-to-spike coded sensors [32], foveated sensors [33], spatial contrast sensors [34]-[35], combined spatio-temporal contrast sensors [36]-[37], and velocity sensors [38]-[39].

One of the limitations of Event-driven hardware compared to Frame-driven equipment is that hardware time-multiplexing is not possible. For example, present day hardware implementations of Frame-driven ConvNets [40] extensively exploit hardware multiplexing by fetching intermediate data in and out between processing hardware and memory. This way, arbitrarily large systems can be implemented by trading off speed. This is not possible in Event-driven hardware, as events need to "flow" and each module has to hold its instantaneous state.

Another disadvantage of Event-driven systems, at least at present, is the lack of efficient, fast training. Spike-Time-Dependent-Plasticity (STDP) [41] seems to be a promising unsupervised learning scheme, but it is slow and requires learning synapses with special properties [42]. Other research efforts are dedicated to algorithmic solutions for supervised STDP type learning [19]-[21]. However at the moment, this field is still quite incipient.

Nevertheless, Event-driven sensing and processing has many attractive features, and research in this direction is certainly worth pursuing.

VII. CONCLUSIONS

A formal method for mapping parameters from a Frame-driven (vision) neural system to an Event-driven system has been presented. Given the extra timing considerations in Frame-free Event-driven systems, extra degrees of freedom become available. This mapping was illustrated by applying it

to example ConvNet systems for recognizing orientations of rotating human silhouettes and fast poker card symbols recorded with real DVS retina chips. The recordings were fed to a hierarchical feed forward spike-driven ConvNet which included 20 Event-driven Convolution modules. The systems were simulated with a dedicated event-driven simulator. The results confirm the high speed response capability of Event-driven sensing and processing systems, as recognition is achieved while the sensor is delivering its output.

VIII. ACKNOWLEDGEMENTS

This work was supported by european CHIST-ERA grant PNEUMA funded by Spanish MICINN (PRI-PIMCHI-2011-0768), Spanish grant (with support from the European Regional Development Fund) TEC2009-10639-C04-01 (VULCANO) and Andalucian grant TIC609 (NANONEURO).

IX. REFERENCES

[1] P. Lichtsteiner, C. Posch, and T. Delbrück, "A 128x128 120dB 30mW asynchronous vision sensor that responds to relative intensity change," *IEEE Int. Solid-State Circ. Conf.* (ISSCC) 2006.

[2] P. Lichtsteiner, C. Posch, and T. Delbrück, "A 128x128 120 dB 15μs latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, No. 2, pp. 566-576, February 2008.

[3] J. Kramer, "An integrated optical transient sensor," *IEEE Trans. on Circ. and Syst., Part II*, vol. 49, no. 9, pp. 612-628, Sep. 2002.

[4] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143dB dynamic range asynchronous address-event PWM dynamic image sensor with lossless pixel.level video-compression," in *IEEE Int. Solid-State Circ. Conf.* (ISSCC) *Dig. of Tech. Papers*, pp. 400-401, Feb. 2010.

[5] J. A. Leñero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco, "A 3.6μs Latency Asynchronous Frame-Free Event-Driven Dynamic-Vision-Sensor," *IEEE J. Solid-State Circuits*, vol. 46, No. 6, pp. 1443-1455, June 2011.

[6] F. Gomez-Rodríguez, R.Paz-Vicente, et al., "AER tools for communications and debugging," *Proc. of the IEEE Int. Symp. on Circ. and Syst.* (ISCAS 2006), pp. 3253-3256, May 2006.

[7] E. Chicca, A. M. Whatley, V. Dante, P. Lichtsteiner, T. Delbrück, P. Del Giudice, R. J. Douglas, and G. Indiveri, "A multi-chip pulse-based neuromorphic infrastructure and its application to a model of orientation selectivity," *IEEE Trans. Circ. and Syst. I,Regular Papers*, vol. 5, no. 54, pp. 981-993, 2007.

[8] R. Serrano-Gotarredona, T. Serrano-Gotarredona, A. Acosta-Jimenez, and B. Linares-Barranco, "A neuromorphic cortical-layer microchip for spike-based event processing vision systems," *IEEE Trans. Circuits and Systems, Part-I: Regular Papers*, vol. 53, No. 12, pp. 2548-2566, December 2006.

[9] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, L. Camuñas-Mesa, R. Berner, M. Rivas, T. Delbrück, S. C. Liu, R. Douglas, P. Häfliger, G. Jiménez-Moreno, A. Civit, T. Serrano-Gotarredona, A. Acosta-Jiménez, B. Linares-Barranco, "CAVIAR: A 45k-Neuron, 5M-Synapse, 12G-connects/sec AER Hardware Sensory-Processing- Learning-Actuating System for High Speed Visual Object Recognition and Tracking," *IEEE Trans. on Neural Networks*, vol. 20, No. 9, pp. 1417-1438, September 2009.

[10] L. Camuñas-Mesa, A. Acosta-Jiménez, C. Zamarreño-Ramos, T. Serrano-Gotarredona, and B. Linares-Barranco, "A 32x32 Pixel Convolution Processor Chip for Address Event Vision Sensors with 155ns Event Latency and 20Meps Throughput," *IEEE Trans. Circ. and Syst.*, vol. 58, No. 4, pp. 777-790, April 2011.

[11] L. Camuñas-Mesa, C. Zamarreño-Ramos, A. Linares-Barranco, A. Acosta-Jiménez, T. Serrano-Gotarredona, and B. Linares-Barranco, "An Event-Driven Multi-Kernel Convolution Processor Module for Event-Driven Vision Sensors," *IEEE J. of Solid-State Circuits*, vol. 47, No. 2, pp. 504-517, Feb. 2012.

[12] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, No. 4, pp. 541.551, 1989.

[13] K. Chellapilla, M. Shilman, and P. Simard, "Optimally combining a cascade of classifiers," in *Proc. of Document Recognition and Retrieval* 13, Electronic Imaging, 6067, 2006.

[14] R. Vaillant, C. Monrocq, and Y. LeCun, "Original approach for the localisation of objects in images," *IEE Proc on Vision, Image, and Signal Processing*, vol. 141, no. 4, pp. 245.250, August 1994.

[15] M. Osadchy, Y. LeCun, and M. Miller, "Synergistic face detection and pose estimation with energy-based models," *Journal of Machine Learning Research*, vol. 8, pp. 1197.1215, May 2007.

[16] C. Garcia and M. Delakis, "Convolutional face finder: A neural architecture for fast and robust face detection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 11, pp. 1408.1423, 2004.

[17] F. Nasse, C. Thurau, and G. A. Fink, "Face detection using gpu based convolutional neural networks," *Lecture Notes in Computer Science, Computer Analysis of Images and Patterns*, vol. 5702/2009, pp. 83.90, 2009.

[18] A. Frome, G. Cheung, A. Abdulkader, M. Zennaro, B. Wu, A. Bissacco, H. Adam, H. Neven, and L. Vincent, "Large-scale Privacy Protection in Google Street View," in *Int. Conf. on Comp. Vision* (ICCV.09), 2009.

[19] S. M. Bohte, J. N. Kok, and H. La Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, 48, pp. 17-38, 2003.

[20] O. Booij, et al., "A gradient descent rule for spiking neurons emitting multiple spikes," *Information Processing Letters*, 95(6), pp. 552-558, 2005.

[21] F. Ponulak and A. Kasinski, "Supervised Learning in Spiking Neural Networks with ReSuMe: Sequence Learning, Classification, and Spike Shifting," *Neural Computation*, Vol. 22, No. 2, pp. 467-510, February 2010.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278-2324, Nov. 1998.

[23] C. Farabet, B. Martini, P. Akserod, S. Talay, Y. LeCun, and E. Culurciello, "Hardware Accelerated Convolutional Neural Networks for Synthetic Vision Systems," *Proc. of the IEEE Int. Symp. Circ. and Systems* (ISCAS), pp. 257-260, 2010.

[24] http://aerst.wiki.sourceforge.net, in preparation.

[25] http://jaer.wiki.sourceforge.net

[26] T. Delbrück and P. Lichtsteiner, "Fast sensory motor control based on event-based hybrid neuromorphic-procedural system," *Proc. of the IEEE Int. Symp. Circ. and Systems* (ISCAS), pp. 845 - 848, 2007.

[27] S. Joshi, S. Deiss, M. Arnold, J. Park, T. Yu, G. Cauwenberghs, "Scalable Event Routing in Hierarchical Neural Array Architecture with Global Synaptic Connectivity," *Int. Workshop on Cellular Nanoscale Networks and their Applications* (CNNA), Feb. 2010.

[28] L. Camuñas-Mesa, J. A. Pérez-Carrasco, C. Zamarreño-Ramos, T. Serrano-Gotarredona, and B. Linares-Barranco, "On Scalable Spiking ConvNet Hardware for Cortex-Like Visual Sensory Processing Systems," *Proc. of the IEEE Int. Symp. on Circ. and Syst.* (ISCAS 2010), pp. 249-252, June 2010.

[29] A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, A. Jiménez, M. Rivas, G. Jiménez, and A. Civit, "On the AER convolution processors for FPGA," *Proc. of the IEEE Int. Symp. on Circ. and Syst.* (ISCAS 2010), pp. 4237-4240, June 2010.

[30] R. Silver, K. Boahen, S. Grillner, N. Kopell and K. L. Olsen, "Neurotech for neuroscience: Unifying concepts, organizing principles, and emerging tools," *Journal of Neuroscience*, vol. 27, no. 44, pp. 11807-819, October 2007.

[31] E. Culurciello, R. Etienne-Cummings, and K. A. Boahen, "A biomorphic digital image sensor," *IEEE J. of Solid-State Circ.*, vol. 38, pp. 281-294, 2003.

[32] S. Chen, and A. Bermak, "Arbitrated time-to-first spike CMOS image sensor with on-chip histogram equalization," *IEEE Trans. on VLSI Systems*, vol. 15, no. 3, 346 - 357. Mar. 2007.

[33] M. Azadmehr, H. Abrahamsen, and P. Hafliger, "A foveated AER imager chip," *Proc. of the IEEE Int. Symp. on Circ. and Syst.* (ISCAS), vol. 3, pp. 2751-2754, 2005.

[34] J.Costas-Santos, T. Serrano-Gotarredona, R. Serrano-Gotarredona and B. Linares-Barranco, "A spatial contrast retina with on-chip calibration for neuromorphic spike-based AER vision systems," *IEEE Transactions on Circuits and Systems, Part I*, vol. 54, no. 7, pp. 1444-58, 2007.

[35] J. A. Leñero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco, "A five-decade dynamic range ambient-light-independent calibrated signed-spatial-contrast AER retina with 0.1ms latency and optional time-to-first-spike mode," *IEEE Trans. on Circ. and Syst. Part I*, vol. 57, no. 10, pp. 2632-2643, Oct. 2010.

[36] K. A. Zaghloul, and K. Boahen, "Optic nerve signals in a neuromorphic chip I: outer and inner retina models," *IEEE Trans. on Biom. Eng.*, vol. 51, no. 4, pp. 657-666, Apr. 2004.

[37] K. A. Zaghloul, and K. Boahen, "Optic nerve signals in a neuromorphic chip II: testing and results," *IEEE Trans. on Biom. Eng.*, vol. 51, no. 4, pp.

667-675, Apr. 2004.
[38] J. Kramer, R. Sarpeshkar, and C. Koch, "Pulse-based analog VLSI velocity sensors," *IEEE Trans. Circuits Syst.* II, vol. 44, pp. 86-101, 1997.
[39] C. M. Higgins and S. A. Shams, "A biologically inspired modular VLSI system for visual measurement of self-motion," *IEEE Sensors Journal*, vol. 2, no. 6, pp. 508-528, Dec. 2002.
[40] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello, "Hardware Accelerated Convolutional Neural Networks for Synthetic Vision Systems," *Proc. IEEE Int. Symp. Circ. and Syst.* (ISCAS10), pp. 257-260, 2010.
[41] T. Masquelier, R. Guyonneau, and S. Thorpe, "Competitive STDP-Based Spike Pattern Learning," *Neural Comp.* vol. 21, pp. 1259-1276, 2009.
[42] G. Indiveri, "Neuromorphic bistable VLSI synapses with spike-timing-dependent plasticity," Advances in Neural Information Processing Systems (NIPS), vol. 15, pp. 1091-1098, 2002.
[43] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello and Y. LeCun, "NeuFlow: A Runtime-Reconfigurable Dataflow Processor for Vision," *Proc. of Embedded Computer Vision Workshop* (ECVW'11), 2011.
[44] C. Farabet, Y. LeCun, and E. Culurciello, "NeuFlow: A Runtime Reconfigurable Dataflow Architecture for Vision," in *Snowbird Learning Workshop*, Cliff Lodge, April 2012.
[45] C. Zamarreño-Ramos, A. Linares-Barranco, T. Serrano-Gotarredona, and B. Linares-Barranco, "Multi-Casting Mesh AER: A Scalable Assembly Approach for Reconfigurable Neuromorphic Structured AER Systems. Application to ConvNets," *IEEE Trans. on Biomedical Circuits and Systems*, in Press.
[46] C. Farabet, R. Paz, J. A. Pérez-Carrasco, C. Zamarreño-Ramos, A. Linares-Barranco, Y. LeCun, E. Culurciello, T. Serrano-Gotarredona, and B. Linares-Barranco, "Comparison between Frame-Constraint Fix-Pixel-Value and Frame-Free Spiking Dynamic-Pixel ConvNets for Visual Processing," *Frontiers in Neuromorphic Engineering*, vol. 6, March 2012, doi: 10.3389/fnins.2012.00032.
[47] J. Poulton, R. Palmer, A. M. Fuller, T. Greer, J. Eyles, W. J. Dally and M. Horowitz, "A 14-mW 6.25-Gb/s Transceiver in 90-nm CMOS," IEEE J. Solid-State Circ., vol. 42, No. 12, pp. 2745-2757, Dec. 2007.
[48] C. Zamarreño-Ramos, T. Serrano-Gotarredona, and B. Linares-Barranco, "An Instant-Startup Jitter-Tolerant Manchester-Encoding Serializer/ Deserializar Scheme for Event-Driven Bit-Serial LVDS Inter-Chip AER Links," *IEEE Trans. Circ. and Syst. Part-I*, vol. 58, No. 11, pp. 2647-2660, Nov. 2011.
[49] C. Zamarreño-Ramos, T. Serrano-Gotarredona, and B. Linares-Barranco, "A 0.35um Sub-ns Wake-up Time ON-OFF Switchable LVDS Driver-Receiver Chip I/O Pad Pair for Rate-Dependent Power Saving in AER Bit-Serial Links," *IEEE Trans. on Biomedical Circuits and Systems*, in Press.
[50] W. Gerstner and W. Kistler, *Spiking Neuron Models. Single Neurons, Populations, Plasticity*, Cambridge University Press, 2002.

José Antonio Pérez-Carrasco received the degree in telecommunication engineering in 2004, and his PhD degree in 2011 from the University of Seville, Sevilla, Spain. He started collaborating with the Biomedical Image Processing Group (BIP) in 2003, when he was working on his B.Sc. Thesis under the supervision of the BIP group leaders Dr. Serrano and Dr. Acha. After working for some companies, in 2005 Dr. Pérez-Carrasco received two 1-year research grants to implement vision processing algorithms within the Sevilla Microelectronics Institute in collaboration with the BIP group at the University of Seville. In 2007 he received a PhD grant and he obtained his doctoral degree in March 2011. He is currently an Assistance Professor at the Dept. of Signal Theory at the University of Seville. His research interests include visual perception, real-time processing, pattern recognition, image processing and its medical applications.

Bo Zhao received the BS and MS degrees in Electronic Engineering from Beijing Jiaotong University, Beijing, China, in 2007 and 2009, respectively. He is currently working toward the PhD degree in the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests are in design and VLSI implementation of bio-inspired vision processing algorithms. He is a student member of the IEEE.

Carmen Serrano received the M.S. degree in Telecommunication Engineering from the University of Seville, Spain, in 1996 and the Ph.D. degree in January 2002. In 1996, she joined the Signal Processing and Communication Department at the same university, where she is currently Tenured Professor. Her research interests concern image processing and, in particular, color image segmentation, classification and compression, mainly with biomedical applications.

Begoña Acha received the Ph.D. degree in Telecommunication Engineering in July 2002. She has been working since 1996 in the Signal Processing and Communications Department of the University of Seville, where she is currently Tenured Professor. Her current research activities include works in the field of color image processing and its medical applications.

Teresa Serrano-Gotarredona received the B.S. degree in electronic physics and the Ph.D. degree in VLSI neural categorizers from the University of Sevilla, Sevilla, Spain, in 1992, and 1996, respectively, and the M.S. degree in electrical and computer engineering from The Johns Hopkins University, Baltimore, MD, in 1997. She was an Assistant Professor in the Electronics and Electromagnetism Department, University of Sevilla from September 1998 until September 2000. Since September 2000, she has been a Tenured Scientist at the National Microelectronics Center, (IMSE-CNM-CSIC), Sevilla, Spain, and in 2008 she was promoted to Tenured Researcher.

Her research interests include analog circuit design of linear and nonlinear circuits, VLSI neural-based pattern recognition systems, VLSI implementations of neural computing and sensory systems, transistor parameters mismatch characterization, address-event-representation VLSI, RF circuit design, nanoscale memristor-type AER, and real-time vision processing chips. She is coauthor of the book *Adaptive Resonance Theory Microchips* (Norwell, MA: Kluwer, 1998).

Dr. Serrano-Gotarredona was corecipient of the 1997 IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS Best Paper Award for the paper "A Real-Time Clustering Microchip Neural Engine." She was also corecipient of the 2000 IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART I: REGULAR PAPERS Darlington Award for the paper "A General Translinear Principle for Subthreshold MOS Transistors." She is an officer of the IEEE CAS Sensory Systems Technical Committee. She is Associte Editor of the IEEE Transactions on Circuits and Systems-Part I, Regular Papers, since December 2011, and Associate Editor for PLoS ONE since 2008.

Shouchun Cheng Shoushun Chen received his B.S. degree from Peking University, M.E. degree from Chinese Academy of Sciences and Ph.D degree from Hong Kong University of Science and Technology in 2000, 2003 and 2007, respectively. He held a post-doctoral research fellowship in the Department of Electronic & Computer Engineering, Hong Kong University of Science and Technology for one year after graduation. From February 2008 to May 2009 he was a post-doctoral research associate within the Department of Electrical Engineering, Yale University. In July 2009, he joined Nanyang Technological University as an assistant professor.

Dr. Chen is a member of IEEE. He serves as a technical committee member of Sensory Systems, IEEE Circuits and Systems Society (CASS); Associate Editor of IEEE Sensors Journal; Associate Editor of Journal of Low Power Electronics and Applications; Program Director (Smart Sensors) of VIRTUS, IC Design Centre of Excellence; Regular reviewer for a number of international conferences and journals such as TVLSI, TCAS-I/II, TBioCAS, TPAMI, Sensors, TCSVT, etc.

Bernabé Linares-Barranco (M'94–F'10) received the B.S. degree in electronic physics, the M.S. degree in microelectronics, and the Ph.D. degree in high-frequency OTA-C oscillator design from the University of Sevilla, Sevilla, Spain, in 1986, 1987, and 1990, respectively, and the Ph.D. degree in analog neural network design from Texas A&M, College Station, in 1991. Since September 1991, he has been a Tenured Scientist at the Sevilla Microelectronics Institute (IMSE), which is one of the institutes of the National Microelectronics Center (CNM) of the Spanish Research Council (CSIC) of Spain. On January 2003, he was promoted to Tenured Researcher and, in January 2004, to Full Professor of Research. Since March 2004, he has also been a part-time Professor with the University of Sevilla. From September 1996 to August 1997, he was on sabbatical stay at the Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD, as a Postdoctoral Fellow. During Spring 2002, he was a Visiting Associate Professor at the Electrical Engineering Department, Texas A&M University.

He is coauthor of the book Adaptive Resonance Theory Microchips (Norwell, MA: Kluwer, 1998). He was also the coordinator of the EU-funded CAVIAR project. He has been involved with circuit design for telecommunication circuits, VLSI emulators of biological neurons, VLSI neural-based pattern recognition systems, hearing aids, precision circuit design for instrumentation equipment, bio-inspired VLSI vision processing systems, transistor parameter mismatch characterization, address-event-representation VLSI, RF circuit design, real-time vision processing chips, and extending AER to the nanoscale.

Dr. Linares-Barranco was a corecipient of the 1997 IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS Best Paper Award for the paper "A Real-Time Clustering Microchip Neural Engine." He was also corecipient of the 2000 IEEE Circuits and Systems (CAS) Darlington Award for the paper "A General Translinear Principle for Subthreshold MOS Transistors.". From July 1997 until July 1999, he was an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II, ANALOG AND DIGITAL SIGNAL PROCESSING, and from January 1998 to December 2009, he was an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS. He is an Associate Editor of Frontiers in Neuromorphic Engineering since May 2010. He was the Chief Guest Editor of the 2003 IEEE TRANSACTIONS ON NEURAL NETWORKS Special Issue on Neural Hardware Implementations. From June 2009 until May 2011, he was the Chair of the Sensory Systems Technical Committee of the IEEE CAS Society. In March 2011 he became Chair of the IEEE CAS Society Spain Chapter.

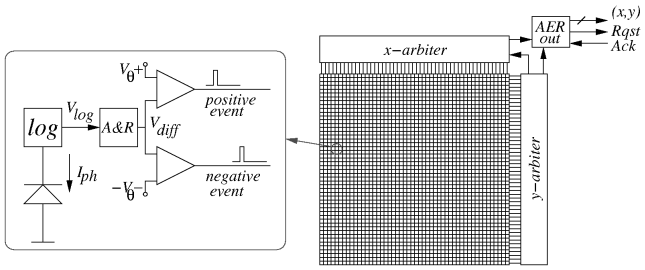


Fig. 1: Typical Event-Driven DVS (Dynamic Vision Sensor) chip structure and pixel diagram

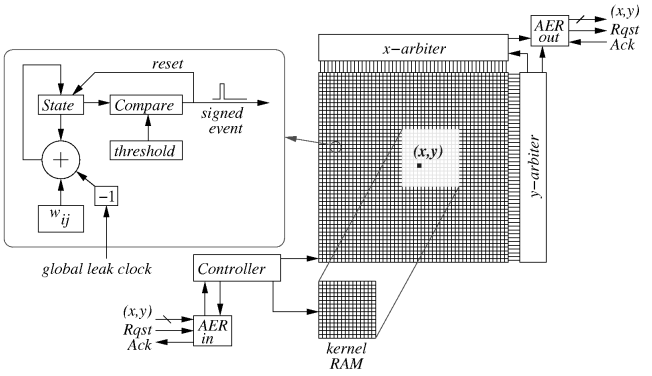


Fig. 2: Typical Event-Driven Convolution Module Chip structure and pixel diagram

I. APPENDIX 1. DVS CAMERA

Several DVS (Dynamic Vision Sensor) cameras have been reported recently [1]-[5]. They all share the schematic operation and structure shown in Fig. 1. Each pixel contains a photosensor. Its photocurrent is transformed into a logarithmic voltage $V_{log} = V_o \log(I_{ph})$. This voltage is fed to an "*Amplify and Reset*" (A&R) block, the output of which is given by

$$V_{diff}(t) = A(V_{log}(t) - V_{log}(t_o)) = AV_o \log\left(\frac{I_{ph}(t)}{I_{ph}(t_o)}\right) \quad (1)$$

where t_o is the time of the previous reset (and event). A new event is generated when V_{diff} reaches one of the thresholds, or equivalently, when $I_{ph}(t)/I_{ph}(t_o)$ reaches either $C^+ = \exp(V_{\theta^+}/AV_o)$ or $1/C^- = \exp(-V_{\theta^-}/AV_o)$. If V_{diff} reaches the positive threshold V_{θ^+} a positive event is sent out, and if it reaches the negative threshold V_{θ^-} a negative event is sent out. Reported DVS sensors can have a contrast sensitivity as low as $C^{\pm 1} = 1{,}1$ (10% contrast sensitivity) [5]. Pixels in the array generate events asynchronously. These are arbitrated by peripheral row and column arbiters [1]-[5],[8]-[11], and sent off chip by writing the (*x*,*y*) coordinates and sign of each pixel event on the off-chip AER bus. The delay between an event being generated in a pixel and written off chip is typically less than $1\mu s$ [5]. Maximum peak event rates of reported DVS chips vary from 1*Meps* [1]-[2] to 20*Meps* [5].

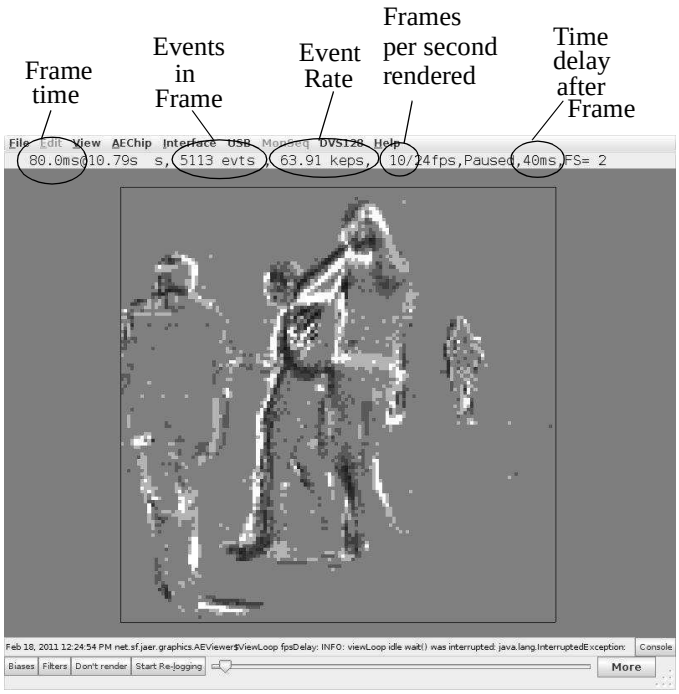


Fig. 3: Example snapshot of event viewer window of jAER. In this example, events are histogrammed into 2D images for time intervals of 80ms. The frame shown contains 5113 events. The actual event rate was 63.91 keps, and frames were rendered at 10 frames per second with a 40ms delay.

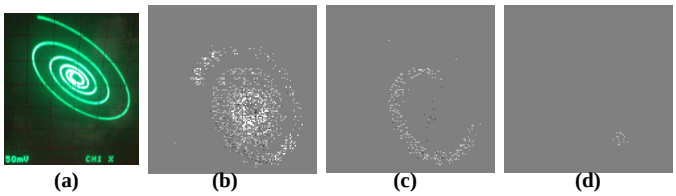
II. APPENDIX 2. AER CONVOLUTION CHIP

Reported AER ConvChips (convolution chips) [8], [10]-[11] compute convolutions event by event. Fig. 2 shows a typical ConvChip floorplan structure. It contains an array of pixels each holding its state. When an input event (*x*,*y*) is received at the input port, the kernel $\{w_{ij}\}$ stored in the kernel RAM is copied around the pixels at coordinate (*x*,*y*), and added/subtracted to/from the respective pixel state. The pixel state is compared against two thresholds, one positive and one negative. If the state reaches one of the thresholds, the pixel sends out a signed event to the peripheral arbiters, and the pixel coordinate with the event sign is written on the output AER bus. In parallel to this event-driven process, there is a periodic leak process common to all pixels. A global leak clock periodically decreases the neurons' states towards their resting level.

Reported AER ConvChips can handle input flows of up to 20*Meps*, produce peak output event rates of up to 45*Meps*, and have kernel sizes of up to 32x32 pixels and pixel array sizes of up to 64x64. They can also be assembled modularly in 2D arrays for processing larger pixel arrays (an *N*x*M* array of 64x64 ConvChips can process input spaces of (*N*x64)x(*M*x64) pixels).

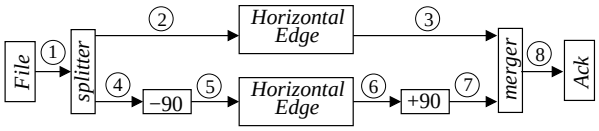
Although at present reported ConvChips contain one convolution module per chip, it is feasible to integrate multiple modules per chip (several tens) together with dedicated event routing and mapping modules [27]-[28],[45]-[46] in modern deep submicron CMOS technologies. Many chips of this type (about a hundred) could be assembled modularly on a PCB, thus hosting thousands of convolutional modules per PCB.

FPGA versions of Event-driven convolution modules are also under development [29], and it is possible to program hundreds of these in a single high end modern FPGA [45].



(a) (b) (c) (d)

Fig. 4: Illustration of event recording time resolution adjustments with jAER. (a) 5KHz rotating spiral photographed on analog phosphor oscilloscope in x-y mode. (b) jAER playback of events recorded with DVS camera by setting frame time to 1.2*ms* (1682 events in frame), (c) to 156μs (306 events in frame), and (d) to 8μs (16 events in frame). Recorded event rate was 2*Meps*.




```
%First, we declare sources to the system
%               SOURCES          SOURCES DATA
sources           {1}               {s1}

%Next, we declare blocks
%NAME               IN-CHANNELS  OUT-CHANNELSPARAMS      STATES
splitter            {1}          {2,4}    {params1}      {state1}
h_sobel             {2}          {3}      {params2}      {state2}
imrotate90          {4}          {5}      {params3}      {state3}
h_sobel             {5}          {6}      {params4}      {state4}
imrotate-90         {6}          {7}      {params5}      {state5}
merger              {3,7}        {8}      {params6}      {state6}
ack_only            {8}          {8}      {params7}      {state7}
```

Fig. 5: (top) Example block diagram of AER system. (bottom) Netlist file describing it.

III. APPENDIX 3. JAER VIEWER

The jAER (java AER) is an open source software tool written in java by T. Delbrück [25] for controlling AER boards and devices, visualizing event flows in 2D screens, recording them on files, playing back recorded files, and performing a variety of operations and filters on these flows. Fig. 3 shows an example snapshot of the jAER viewer window. The viewer collects events during a given frame time to build a 2D image. These 2D images are sequenced on the computer screen. Frame time can be adjusted dynamically by key strokes down to microseconds or even less in case of very high speed recordings. For example, Fig. 4 shows a DVS event recording of a 5*KHz* spiral on an analog oscilloscope, played back at different frame times (1.2*ms*, 156μs, and 8μs).

IV. APPENDIX 4. AERST SIMULATOR

In this simulator a generic AER system is described by a netlist that uses only two types of elements: *instances* and *channels*. An *instance* is a block that generates and/or produces AER streams. AER streams constitute the branches of the netlist in an AER system, and are called *channels*. The simulator imposes the restriction that a *channel* can only connect a single AER output from one instance to a single AER input of another (or the same) instance. This way, channels represent point-to-point connections. To split and/or merge channels, splitters and/or merger instances must be included in the netlist.

Fig. 5 shows an example netlist and its ASCII file netlist description. The netlist contains 7 instances and 8 channels. The netlist description is provided to the simulator through a text file, which is shown at the bottom of Fig. 5. Channel 1 is a source channel. All its events are available a priori as an input file to the simulator. There can be any arbitrary number of source channels in the system. The following lines describe each of the instances, one line per instance in the network. The first field in the line is the instance name, followed by its input channels, output channels, name of structure containing its parameters, and name of structure containing its state. Each instance is described by a user-made function bearing the name of the instance. The simulator imposes no restriction on the format of the parameter and state structures. This is left open to the user writing the function code of each instance.

Channels are described by two dimensional matrices. Each row in the matrix corresponds to one event. Each row has six components

$$[x, y, \text{sign}, T_{preRqst}, T_{Rqst}, T_{Ack}] \quad (2)$$

'x' and 'y' represent the coordinates or addresses of the event and 'sign' shows its sign. These 3 parameters are the "event parameters", and they are defined by the user. The simulator only transports them through the channels, but does not interpret them. We always use *x,y*, and sign, but these can be freely changed by the user, as long as the instances can interpret them appropriately. The last 3 fields are event timing parameters, and are managed by both the simulator and the instances. Time '$T_{preRqst}$' represents the time at which the event is created at the emitter instance, 'T_{Rqst}' represents the time at which the event is received by the receiver instance, and 'T_{Ack}' represents the time at which the event is finally acknowledged by the receiver instance. We distinguish between a pre-Request time $T_{preRqst}$ and an effective Request time T_{Rqst}. The first is dependent only on the emitter instance, while the second requires the receiver instance to be ready to process an event request (i.e., not to be busy processing a prior event). This way, a full list of events described only by their addresses, sign, and $T_{preRqst}$ times can be provided as source. Once the events are processed by the simulator, their final effective request and acknowledge times are established.

Initially, all times T_{Ack} are set to '-1' to label them as "unprocessed" events. The simulator looks at all channels, selects the earliest (smallest $T_{preRqst}$) unprocessed event, calls the instance of the corresponding channel and transfers the event parameters to it. The instance returns T_{Rqst} and T_{Ack}, updates its internal state, and eventually writes one or more new unprocessed output events on its output channels. The simulator then searches all the channels again for the next unprocessed event. This process continues until there are no more unprocessed events left. The result of the simulation is a list of timed events on all channels. These lists are the flow of events that would have been produced by a real system, and are displayed by the jAER tool.

V. APPENDIX 5. PARAMETER OPTIMIZATION BY SIMULATED ANNEALING

The timing and threshold parameters of a spiking ConvNet can be optimized. For this we called from within MATLAB the AERST simulation program running different input sequences and providing the timing and threshold parameters for each run. We then used the simulated annealing routine within the MATLAB optimization toolbox to find optimum sets for the timing and threshold parameters. The cost function to be minimized by the optimization routine was defined as

$$COST = aN_{FE} - bN_{CE} + P \quad (3)$$

Table 1

optimized parameters											*derived parameters*								*performance*				
T_{R1} (*ms*)	T_{R3} (*ms*)	T_{R5} (*ms*)	th_1	th_3	th_5	th_6	LR_1 (s^{-1})	LR_3 (s^{-1})	LR_5 (s^{-1})	LR_6 (s^{-1})	g_{R3}	g_{R5}	T_{C1} (*ms*)	T_{C3} (*ms*)	T_{C5} (*ms*)	p_1	p_3	p_5	*succ.* (%)	*avg delay* (*ms*)	*min delay* (*ms*)	*avg# events*	*min# events*
0.10	0.50	10.00	0.60	1.00	5.00	1.50	10.00	5.00	2.00	1.00	5.00	20.00	5.00	25.00	100	50.0	50.0	10.0	96.47	40.88	8.41	609	91
1.23	0.61	11.73	0.59	1.00	4.44	1.39	6.48	8.15	13.84	10.44	0.50	19.15	46.45	23.17	100	37.8	37.8	8.5	98.37	51.85	23.50	803	249
1.30	0.79	11.75	0.63	0.96	4.53	1.36	7.79	8.18	13.56	10.40	0.61	14.85	48.45	30.53	100	37.2	38.6	8.5	98.19	59.40	26.83	903	279
1.47	1.34	12.12	0.72	0.94	4.34	1.21	6.48	8.12	13.18	10.72	0.91	9.01	49.73	48.11	100	33.8	35.8	8.3	97.28	49.31	9.31	695	163
1.48	1.34	12.27	0.58	0.95	4.48	1.21	6.21	7.94	13.58	10.52	0.90	9.16	51.50	48.89	100	34.7	36.5	8.2	97.70	42.71	9.26	659	121
1.36	0.84	10.88	0.71	0.99	4.45	1.26	6.83	8.01	13.75	10.60	0.61	13.02	54.95	34.15	100	40.4	40.9	9.2	98.31	43.94	20.43	654	215
1.24	0.85	10.81	0.71	0.98	4.44	1.32	6.83	8.02	13.75	10.60	0.68	12.77	50.26	34.77	100	40.4	41.1	9.3	99.61	43.21	1.93	645	42
0.51	0.90	10.92	0.68	1.00	4.44	1.45	6.80	7.86	13.80	10.78	1.78	12.12	20.60	36.64	100	40.6	40.7	9.2	97.94	48.37	12.02	729	101

Table 2

optimized parameters											*derived parameters*								*performance*				
T_{R1} (*ms*)	T_{R3} (*ms*)	T_{R5} (*ms*)	th_1	th_3	th_5	th_6	LR_1 (s^{-1})	LR_3 (s^{-1})	LR_5 (s^{-1})	LR_6 (s^{-1})	g_{R3}	g_{R5}	T_{C1} (*ms*)	T_{C3} (*ms*)	T_{C5} (*ms*)	p_1	p_3	p_5	*succ.* (%)	*avg delay* (*ms*)	*min delay* (*ms*)	*avg# events*	*min# events*
0.00	0.10	0.46	0.64	1.42	7.36	2.17	0.72	0.90	1.21	0.72	-	4.60	0	8.00	5.00	0	80	12	91.57	5.97	0.20	30.85	14
0.00	0.07	0.24	0.40	1.55	7.02	3.39	13.01	4.14	8.32	2.17	-	3.43	0	10.23	5.00	0	146	21	90.10	6.02	2.22	31.80	14
0.00	0.08	0.44	0.68	1.18	8.11	2.17	13.83	3.04	7.96	4.94	-	5.50	0	7.37	5.00	0	92	11	91.43	5.63	0.80	31.45	13
0.00	0.19	0.57	0.38	3.99	3.62	1.82	6.37	6.60	5.96	5.11	-	3.00	0	6.03	5.00	0	32	8.8	91.36	4.68	0.33	30.60	11
0.00	0.14	0.54	0.33	4.20	3.60	1.82	8.72	6.60	5.90	5.13	-	3.86	0	4.66	5.00	0	33	9.3	90.35	4.26	0.12	34.20	12

where N_{FE} is the number incorrect positive events, N_{CE} is the number of correct positive events, $a = 1$, $b = 0.1$, and $P = 10^4$ if for some category there is no output event (neither positive nor negative). Otherwise $P = 0$. With such high value for *P*, the optimization discards directly those sets of parameters that do not yield any output event for a category. Then, if there are output events for all categories, the process penalizes if there are incorrect positive events. It also tries to maximize (with a weaker weight) the number of correct positive events.

Table 1 illustrates some optimized results for the experiment on human silhouettes detection. Each row corresponds to a different optimum, while the top row corresponds to the heuristic case described in Section IV, for comparison. Table 1 shows the "optimized parameters", the "derived parameters" (T_{C5} was always set to 100*ms*), and the following "performance" results: (a) success rate, (b) delays (average and minimum) and (c) number of events to first correct recognition (average and minimum). As can be seen, the optimizations produced parameters similar to our "intuition" for all thresholds, refractory times T_{R3}, T_{R5}, and leak rates L_{R1}, and L_{R3}. However, parameters T_{R1}, L_{R5}, and L_{R6} were optimized to quite different values. As a result, derived parameters g_{R3} and T_{C1} differ, but parameters g_{R5}, T_{C3}, p_1, p_3, p_5 were quite similar. On the other hand, success rate was improved for the optimized cases as seen in Table 1, although speed response was degraded. The number of events required to achieve first recognition was on average between 600 and 900 events, captured during 40 to 60*ms*. For some transitions and optimizations, the minimum number of events fell to less than 100 events, with delays around 10*ms* and below. This is due to the statistical distribution of the first event front, after a transition (if first front events are distributed more uniformly over the entire silhouette, then recognition is faster).

Table 2 illustrates some optimized results for the experiment on poker cards symbol detection. In this case reality moves much faster. Consequently, timing parameters were optimized to quite different values. We intentionally set refractory time of the first layer to zero to allow for maximum speed. Parameter T_{C5} was set to 5*ms* because we observed that this is approximately the maximum time one could set to reconstruct a reasonable good looking frame for a symbol. Recognition rate for this experiment was in the range 90.1-91.6%, which is lower than for the previous experiment on human silhouettes. The obvious reason is that now not only reality is much faster, but the sensor provides also many more noise events, and events are also much more sparse.

When comparing the timing parameters in Table 1 and Table 2 for the optimized event driven networks with the results shown in Table 3 and Table 4 of the main text for the frame driven networks, it is interesting to notice the following observation. The refractory time of the latest layer T_{R5} provides a sense for the timing of the whole network. We can see that for the silhouettes experiment T_{R5} is in the range of 10-12ms, while for the cards experiment it is in the range of 0.3-0.6ms. On the other hand, for the frame driven experiment results shown in Table 3 and Table 4, we can see the optimum frame times were about 50-75ms for the silhouettes and about 2-3ms for the cards. Thus, there seems to be a factor of about 5 between the T_{R5} optimum parameters and the optimum frame times in Table 3 and Table 4. However, this conclusion is quite speculative at this point and needs to be verified for more varied experiments.

VI. APPENDIX 6: COMPARISON OF FRAME-DRIVEN VS. EVENT-DRIVEN HARDWARE REALIZATIONS

In order to compare Frame-Driven vs. Event-Driven hardware ConvNet performance we rely on some reported example realizations and comparisons [43]-[46], as well as

	Present State-of-the-art			Future Outlook	
	Frame-driven	Frame-driven	Event-Driven	Frame-driven	Event-Driven
	nVidia GTX480 GPU	Virtex6 Purdue/NYU	Virtex6 IMSE/US	3D ASIC Purdue/NYU	Grid 40nm
Input scene size	512x512	512x512	128x128	512x512	512x512
Delay	2.7ms/frame	5.5ms/frame	3μs/event	1.3ms/frame	10ns/event
Gabor array	16 convs 10x10 kernels	16 convs 10x10 kernels	64 convs 11x11 kernels	16 convs 10x10 kernels	100 convs 32x32 kernels
Neurons	$4.05x10^6$	$4.05x10^6$	$2.62x10^5$	$4.05x10^6$	10^8
Synapses	$4.05x10^8$	$4.05x10^8$	$3.20x10^7$	$4.05x10^8$	10^{11}
Conn/s	$1.6x10^{11}$	$7.8x10^{10}$	$2.6x10^9$	$3.0x10^{11}$	$4.0x10^{13}$
Power	220W	10W	10W	3W	0.25W @ 10Meps/chip

Table 3. Hardware Performance Comparison of present and futuristic projections on Frame vs. Event-Driven Realizations

some futuristic projections [46]. Table 3 summarizes hardware performance figures for three implementations. The first three columns correspond to present day state-of-the-art while the last two are for futuristic projections. All of them correspond to implementing a set of Gabor convolution filters. The frame-driven cases correspond to implementing 16 Gabor filters with kernel of size 10x10 pixel operating on input images of size 512x512 pixels. For the frame-driven examples speed is expressed in frames per second. The GPU realization is very fast but consumes a high power, while the Virtex6 realization [43]-[44] is comparable in speed but consumes twenty times less power. The futuristic projection of this same system on an ASIC using IBM 65nm 3D technology improves speed by about five times and power a factor of three. Recently, an event-driven configurable ConvNet array implemented on Virtex6 is reported [45]-[46] (3rd column in Table 3). Speed is expressed in terms of computation delay per event. Since event-driven systems are pseudo-simultaneous, the filtered output is available as soon as enough representative input events are received. Therefore, speed response is not determined by the delay of the hardware, but by the statistical timing distribution of the events provided by the sensor. The power dissipation of a system implemented in an FPGA is mainly determined by FPGA clock and resources used. Since the frame-driven and event-driven Virtex6 realizations in the 2nd and 3rd columns of Table 3 have the same clock and use similar FPGA resources, power consumption is also similar.

A futuristic projection of an event-driven AER convolution chip fabricated in 40nm technology can be estimated to host one million neurons and one giga synapses [11], while processing events with delays of 10ns. Assembling AER grids [45] of such chips with 100 chips per PCB (Printed Circuit Board) results in the estimated performance of the last column in Table 3. AER processing chips power consumption has two components, one that scales with the number of events processed per unit time, plus another background component which is constant. A great amount of power is consumed by inter-chip event communication. Using modern low power serial drivers [47] that consume 2.25mW at 6.25Gb/sec, combined with circuit techniques that scale power with event rate [48]-[49] it would be feasible to reach a power consumption figure of 115μW at 10Meps (mega events per second) event communication rate per link. If each chip has four such links in a grid, communication power per chip would be about 0.5mW. Assuming in-chip event-rate dependent power is twice and background power is similar, each chip could consume about 2.5mW when generating 10Meps. A grid of 100 of those chips would dissipate 250mW.

VII. APPENDIX 7: REVERSE CORRELATION RECONSTRUCTIONS

In order to characterize the internal representations and dynamics of the resulting event driven networks, we show here reverse correlation reconstructions [50] for the output layer neurons as well as for the layer C5 neurons (see Fig. 9 of main paper). The purpose of reverse correlation reconstruction is to discover what type of spatio-temporal patterns (at the input or at intermediate feature maps) elicit activation of a given neuron. For example, if in the poker cards experiment we pick the output neuron for category ‘heart’, we follow the next steps: (1) for all trials of all test simulations, we pick the full list of ‘heart’ category output events $\{o_{heart}(k)\}$ that were positive; (2) for a given time window T_w before each output event k, we look at the neurons j we are interested in, and count for each the number of positive $np_{jk}(T_w)$ and negative $nn_{jk}(T_w)$ events that occurred during this time window; (3) and compute for each of these neurons of interest j the following reconstruction value

$$r_j(T_w) = \frac{\sum_{k=1}^{K} (np_{jk}(T_w) - nn_{jk}(T_w))}{KT_w} \quad (4)$$

This number represents the average number of events (activity) of a pixel j that would contribute to trigger one event at the output neuron representing category ‘heart’. This average number is also normalized with respect to the size of the time window T_w, so that one can compare graphically short time windows with respect to larger time windows. Note that, as the T_w increases, the number of events also tends to increase.

Fig. 6(a-d) shows the reverse correlation reconstruction for all four output category neurons (Layer C6), and Fig. 6(e-l) for

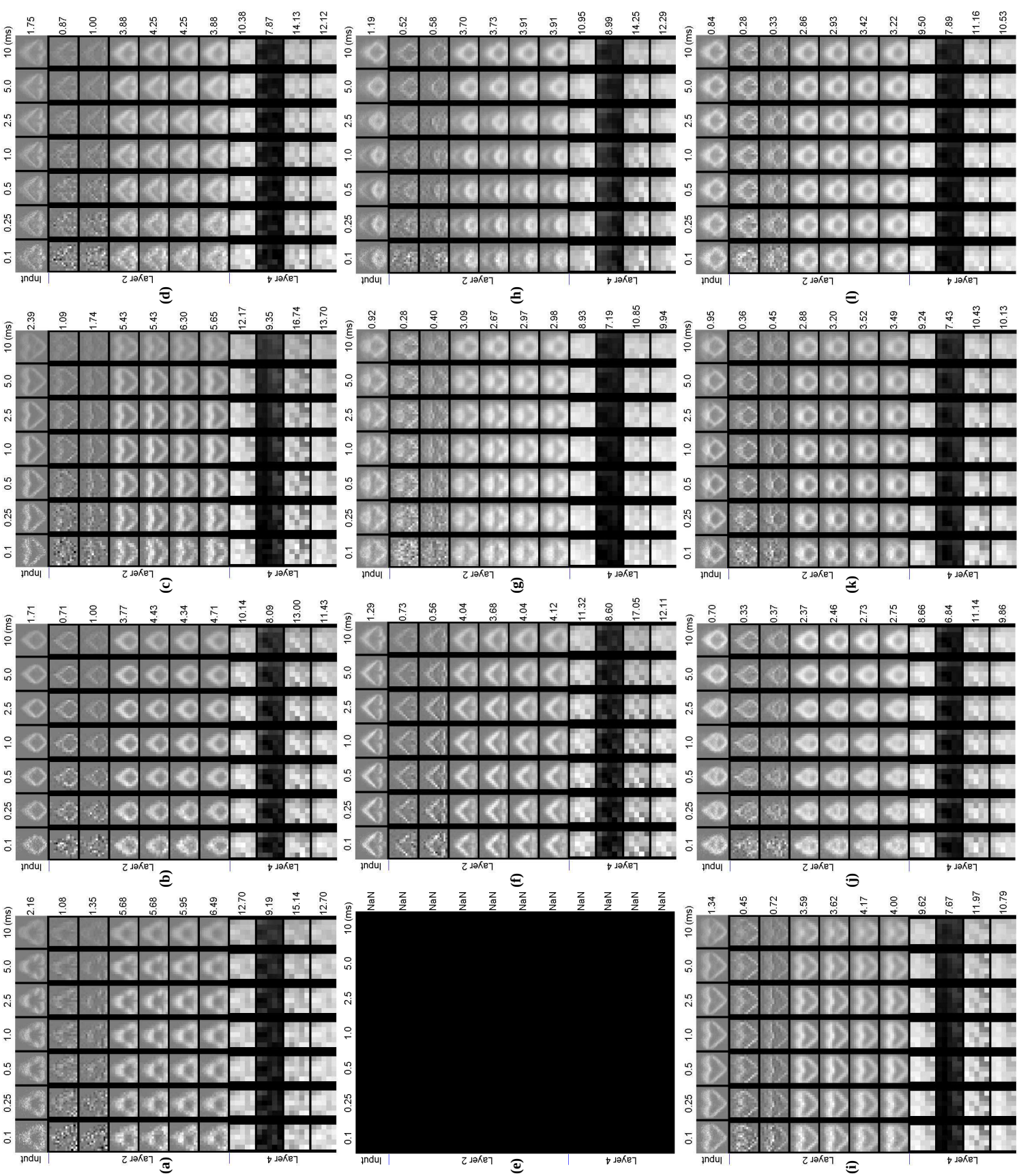


Fig. 6: Reverse correlation reconstruction of input and feature map patterns for output Layer C6 neurons (a-d) and Layer C5 neurons (e-l). Each subfigure shows a mosaic of 7 columns and 11 rows. Each of the 7 columns corresponds to a different time window T_w = {0.1, 0.25, 0.5, 1.0, 2.5, 5.0, 10.0}ms. The 11 rows of each subfigure are as follows: top row corresponds to the 32x32 pixel input pattern, the next 6 rows correspond to the 6 14x14 pixel Feature Maps of Layer S2, and the next 4 rows correspond to the 4 5x5 pixel Feature Maps of Layer S4. To the right of each row the number indicates the maximum absolute value of all pixels in this Feature Map row for all time windows T_w, which has been mapped to either the brightest white or darkest black. Value 0 is always mapped to the same central gray intensity.

all eight Layer C5 neurons. Each Layer C6 or C5 neuron reconstruction corresponds to one subfigure. Each subfigure has 7 columns and 11 rows. Each column corresponds to a different value of the time window T_w , taking the values T_w = {0.1, 0.25, 0.5, 1.0, 2.5, 5.0, 10.0}ms. The first top row corresponds to the 32x32 pixel input. The next six rows correspond to the six 14x14 pixel Feature Maps of Layer S2 (see Fig. 9 of main paper). And the next four rows show the reconstructions of the four 5x5 pixel Feature Maps of Layer S4. The gray level coding is common for each row. Central gray corresponds to $r_j(T_w) = 0$. The number to the right of each row is the maximum absolute value of $r_j(T_w)$ among all neurons of all Feature Maps in that row. This maximum value is corresponds to brightest white or darkest black greyscale value for this row gray level coding.

As can be seen in all subfigures, as the time window T_w increases, the values of the reconstructions $r_j(T_w)$ tend to decrease slightly. We can see that the correct input patterns can be already seen for time windows of T_w = 0.1ms, and seem to be optimum for T_w between 1ms and 2ms.

Another observation is that the full subfigure Fig. 6(e) is empty: no positive events were elicited during all trials for the first neuron of Layer C5. This neuron only produced negative events. This means that this neuron specialized to become negative when detecting the presence of specific features. Similarly, note that the neurons in the second Feature Map of Layer S4 have also specialized on negative events mainly.