HFirst: A Temporal Approach to Object Recognition

Garrick Orchard, Cedric Meyer, Ralph Etienne-Cummings, *Fellow, IEEE*,
Christoph Posch, *Senior Member, IEEE*, Nitish Thakor, *Fellow, IEEE*, and Ryad Benosman

Abstract—This paper introduces a spiking hierarchical model for object recognition which utilizes the precise timing information inherently present in the output of biologically inspired asynchronous address event representation (AER) vision sensors. The asynchronous nature of these systems frees computation and communication from the rigid predetermined timing enforced by system clocks in conventional systems. Freedom from rigid timing constraints opens the possibility of using true timing to our advantage in computation. We show not only how timing can be used in object recognition, but also how it can in fact simplify computation. Specifically, we rely on a simple temporal-winner-take-all rather than more computationally intensive synchronous operations typically used in biologically inspired neural networks for object recognition. This approach to visual computation represents a major paradigm shift from conventional clocked systems and can find application in other sensory modalities and computational tasks. We showcase effectiveness of the approach by achieving the highest reported accuracy to date (97.5% $\pm$ 3.5%) for a previously published four class card pip recognition task and an accuracy of 84.9% $\pm$ 1.9% for a new more difficult 36 class character recognition task.

Index Terms—Neuromorphic computing, computer vision, object recognition, neural nets

---

1 INTRODUCTION

THIS paper tackles the problem of object recognition using a hierarchical spiking neural network (SNN) structure. We present a model developed for object recognition, which we have called HFirst. The name arises because the approach extensively relies on the first spike received during computation to implement a non-linear pooling operation, which is typically required by frame-based convolutional neural networks (CNNs).

We rely on the biological observation that strongly activated neurons tend to fire first [1], [2]. In particular, we focus on the relative timing of spikes across neurons, namely the order in which neurons fire. We will argue that such a scheme allows us to derive temporal features that are particulary suited for robust and rapid object recognition at a very low computational cost. Existing work on artificial neural networks (NNs) tend to assume a predetermined timing which is completely independent of the processing taking place. This prohibits these artificial NNs from using time in their computation. However, the timing of communication (spikes) in biological networks is known to be very important. Much like biological networks, in this paper we exploit spike timing to our advantage in computation. More specifically we rely on the time at which a spike is received to implement a simple non-linear operation which replaces the more computationally intensive maximum operation typically used in non-spiking neural networks for visual processing.

Artificial neural networks, of which CNNs are a subset, have successfully been used in many applications, including signal and image processing [3], [4], and pattern recognition [5], while hardware acceleration of such models allows real-time operation on megapixel resolution video [6]. Although CNN models are argued to be biologically inspired, their artificial implementations are typically far removed from biological neural networks, most of which consist of spiking neurons.

Spiking neural networks have received a lot of attention recently as new, more efficient computing technologies are sought as conventional CMOS technology approaches its fundamental limits. SNNs have the potential to achieve incredibly high power efficiency. This is not a claim that we provide our own evidence for, but is rather based on observations of power consumption in biology (the human brain consumes only 20 W) and recent works which present SNNs on chip with impressive power efficiency. Examples include Neurogrid [7] and IBMs TrueNorth [8] which can simulate 1 million spiking neurons while consuming under 100 mW. In this paper we address the question of how SNNs can be used for visual object recognition.

Modern reconfigurable custom SNN hardware platforms can implement hundreds of thousands to millions of spiking neurons in parallel. Examples of these hardware implementation projects include the integrate and fire array transceiver (IFAT) [9], Hierarchical AER-IFAT [10], Brain Scales [11], spiking neural network architecture (SpiNNaker) [12], Neurogrid [7], Qualcomm's Zeroth Processor, and IBM's TrueNorth [8] (fabricated with Samsung).

- *G. Orchard and N. Thakor are with the Singapore Institute for Neurotechnology (SINAPSE), National University of Singapore, Singapore. E-mail: {garrickorchard, eletnv}@nus.edu.sg.*
- *C. Meyer, C. Posch, and R. Benosman are with the Institut De la Vision, Universite Pierre et Marie Curie, Paris, France. E-mail: meyer.cece@gmail.com, cposch@yahoo.com, ryad.benosman@upmc.fr.*
- *R. Etienne-Cummings is with the Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD. E-mail: retienne@jhu.edu.*

In parallel with these hardware platforms, software platforms for neural computation have emerged, including the neural engineering framework (NEF) [13], Brian [14], and PyNN [15], many of which can be used to configure the hardware platforms previously mentioned. Continued interest and funding from the European Union's Human Brain Project [16] and the USA's brain research through advancing innovative neurotechnologies (BRAIN) project [17] will drive development of such systems for years to come.

As neural simulation hardware matures, so must the algorithms and architectures which can take advantage of this hardware. However, it does not necessarily make sense to directly convert existing computer vision models and algorithms (which process traditional frame based data) to SNN implementation. A central concept within SNNs is that spike timing encodes information, but frames do not contain precise timing information. The timing of the arrival of frames is purely a function of the front end sensor and is completely independent of the scene or stimuli present. In order for a SNN to exploit precise timing, it must operate on data which contains precise timing information and not spike timings artificially generated from frame-based outputs. To obtain visual data with precise timing, we turn our attention to asynchronous AER vision sensors, sometimes referred to as "silicon retinae" [18], [19]. These sensors more closely match the operation of biological retina and do not utilize frames.

Asynchronous AER vision sensors have seen much improvement since their introduction in the early 1990s by Mahowald [20]. Modern change detection AER sensors reliably provide information on changes of illumination at the focal plane over a wide dynamic range and under a variety of lighting conditions. The pixels within such sensors each contain a circuit which continuously performs local analog computation to detect the occurrence and time of changes in intensity for that particular pixel. This computation at the focal plane is a form of redundancy suppression, ensuring that pixels only output data when new information is present (barring some background noise). Furthermore, the time of arrival of data from the sensor accurately represents when the intensity change occurred. Under test conditions sub-microsecond accuracy is achieved, versus accuracy on the order of milliseconds for fast frame-based cameras. This temporal accuracy provides precise spike timing information which can be exploited by a SNN.

Much like SNNs are a more accurate approximation of biological processing hardware, AER vision sensors are a more accurate approximation of the biological retina. The single bit of data provided by a pixel can be likened to a neural spike, and much like a biological retina, the AER sensor performs computation at the focal plane. Notable examples of spiking AER vision sensors include the earliest examples of spiking silicon retinae by Culurciello et al. [18] and Zaghloul and Boahen [19], as well as the more recent dynamic vision sensor (DVS) from Delbruck et al. [21], the sensitive DVS from Serrano-Gotarredona and Linares-Berranco [22], and the asynchronous time-based image sensor (ATIS) from Posch et al. [23]. Operation of these sensors will be discussed in Section 2. For a review of asynchronous event-based vision sensors see Delbruck et al. [24].

With the emergence of these asynchronous vision sensors, many researchers have taken an interest in processing their data in a manner which takes advantage of the asynchronous, high temporal resolution, and sparse representation of the scene they provide. Models of early visual area V1, including saliency, attention, foveation, and recognition [25], [26], [27] have been implemented by combining the reconfigurable IFAT system [28] with the Octopus silicon retina [18]. More recent focuses in the field include stereo vision [29], [30], [31], motion estimation [32], [33], tracking [34], and more object recognition works [35], [36], [37]. Further information on neuromorphic sensory systems can be found in Liu and Delbruck [38].

In this paper we focus on the task of object recognition. The most similar recent works include a VLSI implementation of the HMAX model [39], [40] for recognition which uses spiking neurons throughout [41]. The VLSI spiking HMAX implementation computes all the functions required by HMAX, but operates on 24×24 pixel images, limited by the number of available neurons, and does not run real-time. Adaptations of frame-based CNN techniques for training SNNs and implementing them in FPGA have also been recently presented [42], including a recent PAMI paper [35] which presented a high speed card pip recognition task which we also tackle in this paper as a comparison to existing works.

In this paper we present our SNN architecture dubbed "HFirst", which takes advantage of timing information provided by AER sensors. A key aspect is that our architecture uses spike timing to encode the strength of neuron activation, with stronger activated neurons spiking earlier. This enables us to implement a MAX operation using a simple temporal Winner-Take-All (WTA) rather than performing a synchronous MAX operation as is typically done in frame-based algorithms [39]. Unlike the frame-based MAX operation, which outputs a number representing the strength of the strongest input, the temporal WTA can only output a spike, but by responding with low latency to its inputs, the temporal WTA preserves the time encoding of signal strength. It should be noted that other methods of implementing a MAX operation in spikes have been presented previously [27].

Masquelier and Thorpe [43] also use a temporal WTA, but their approach focuses on static images and spike generation from these images is artificially simulated, whereas we use AER vision sensors [21], [22], [23] to directly capture data from dynamic scenes for recognition. Additionally, Masquelier and Thorpe require their network to be reset before a second object can be recognised, whereas HFirst operates on streaming "video" and can recognise multiple objects in sequence, or even simultaneously.

The HFirst model described here can be used with many of the available AER change detection sensors, and could be implemented on one of many neural processing platforms. For this particular work we analysed HFirst in simulation using a combination of C and Matlab on a desktop PC. Once simulated, the SNN was implemented in real-time on a Xilinx Spartan 6 XC6SLX150-2 FPGA.

The rest of this paper is organized as follows. In the next section we briefly describe the event-based vision sensors, then we describe the neuron model using spike timing for computation in Section 3. The HFirst architecture is described in Section 4, followed by brief analysis of the required computation and real-time implementation.
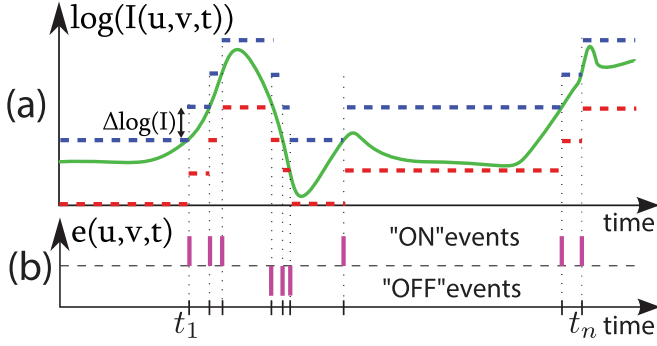
Fig. 1. Event-based vision sensor acquisition principle. (a) typical signal showing the log of luminance of a pixel located at $[u,v]^T$. Dotted lines show how the thresholds for detecting increases and decreases in intensity change as outputs are generated. (b) asynchronous temporal contrast events generated by this pixel in response to the light variation shown in (a).

Testing and results are then presented to showcase the model accuracy before wrapping up with discussions and conclusions.

2 ASYNCHRONOUS CHANGE DETECTION VISION SENSORS

Neuromorphic, event-based vision sensors are a novel type of vision sensor driven by changes within the visual scene, much like the human retina, and differs from conventional image sensors which use artificial timing to control information acquisition. The sensors used in this paper [21], [22], [23] consist of autonomous pixels, each asynchronously generating spike events that encode relative changes in illumination. These sensors capture visual information at a much higher temporal resolution than conventional vision sensors, achieving accuracy down to sub-microsecond levels under optimal conditions. Moreover, since the pixels only detect temporal changes, temporally redundant information is not captured or communicated, resulting in a sparse representation of the scene. Captured events are transmitted asynchronously by the sensor in the form of continuous-time digital words containing the address of the activated pixel using the AER protocol [20].

To better understand the operation of these sensors we will briefly provide a formulation to approximate the sensor response to visual stimuli. Let us define $I(u,v,t)$ as the intensity of a pixel located at $[u,v]^T$, where u and v are spatial co-ordinates in units of pixels. Each pixel of the sensor asynchronously generates events at the precise time when change in the log of the pixel illumination $\Delta log(I(u,v,t))$ is larger than a certain threshold ΔI since the last event, as shown Figs. 1a and 1b. The logarithmic relation means the pixels respond to percentage changes in illumination rather than the absolute magnitude of the change. This allows pixels to operate over a very wide dynamic range ($\geq$120 dB).

Under constant scene illumination the intensity changes seen by the sensor are due to the combination of a spatial image gradient and a component of image motion along that gradient. As described by the equation below which is a first order approximation of the image constancy constraint
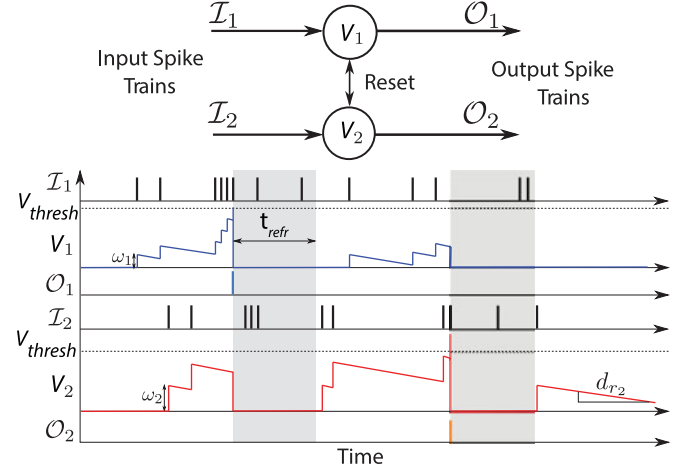

Fig. 2. Operation of an Integrate-and-Fire neurons used, showing how synaptic weights and time affect the neuron membrane potential, as well as the operation of lateral reset connections (lateral meaning connecting to other neurons in the same layer) and refractory period.

$$\frac{dI(u,v,t)}{dt} = -\frac{dI(u,v,t)}{du}\frac{du}{dt} - \frac{dI(u,v,t)}{dv}\frac{dv}{dt}, \qquad (1)$$

where $I(u,v,t)$ is intensity on the image plane, and u and v are horizontal and vertical coordinates measured in units of pixels. The sensor will therefore generate the most events at locations where a large image gradient is present, as will be discussed further in Section 3.2.

3 COMPUTING WITH NEURONS

3.1 Neuron Model

The neuron model we use is a simple Integrate-and-Fire neuron (IF neuron) [44] with linear decay and a refractory period, as shown in Fig. 2. We foresee that the model would translate to hardware implementations which model many neurons in parallel, but the neurons in such hardware implementations may have very limited precision. To account for the possible limited precision in implementation, in software we simulate subthreshold membrane potential decay with 1 ms time precision and restrict all neuron parameters (V_{thresh}, $\frac{I_l}{C_m}$, and t_{refr} in Table 1) to be unsigned 8 bit integers with 1 least significant bit (LSB) corresponding to 1 unit shown in Table 1. During simulation, membrane potential is stored as an integer value in units of millivolts.

The simple behaviour of IF neurons ensures that an output spike can only be elicited by an excitatory input spike, and not by subthreshold membrane potential dynamics in the absence of excitatory input. When an input to a neuron arrives, the neuron's new state (membrane potential) can be

TABLE 1
Neuron Parameters

Layer	V_{thresh}	I_l/C_m	t_{refr}	Kernel Size	Layer Size
S1	200	50	5	7×7×1	128×128×12
C1	1	0	5	4×4×1	32×32×12
S2	100-200	10	10	8×8×12	32×32×N_y
C2	1	0	10	32×32×1	1×1×N_y
unit	mV	mV/ms	ms	synapses	neurons

entirely determined by the time since it was last updated, and its state after the previous update. We therefore need only update a neuron when it receives an input spike (rather than at a constant time interval). Neurons are organized into a hierarchical structure consisting of layers. When an input spike arrives from a lower layer, the update procedure for the neuron is:

if $t_i - t_{lastspike} < t_{refr}$ **then**
    $Vm_i \leftarrow Vm_{i-1}$
else
$$Vm_i \leftarrow \begin{cases} \max\{Vm_{i-1} - \frac{I_l}{C_m}(t_i - t_{i-1}), 0\} & \text{if } Vm_{i-1} \geq 0 \\ \min\{Vm_{i-1} + \frac{I_l}{C_m}(t_i - t_{i-1}), 0\} & \text{if } Vm_{i-1} < 0 \end{cases}$$
    $Vm_i \leftarrow Vm_i + \omega_i$
end if
if $Vm_i \geq V_{thresh}$ **then**
    $Vm_i \leftarrow 0$
    $t_{lastspike} \leftarrow t_i$
    Do(Generate Output Spike)
end if

where t_i is the time at which the ith input spike arrives, $t_{lastspike}$ is the time at which the current neuron last generated an output spike, t_{refr} is the refractory period of the neuron, Vm_i is the membrane voltage after the ith input spike, I_l is the leakage current, C_m is the membrane capacitance, ω_i is the input weight of the ith input spike, and V_{thresh} is the threshold voltage for the current neuron.

Output spikes from a neuron feed similarly into the layer above, but can also affect neurons within the same layer through lateral connections. When an input is received from a lateral connection, it forces the receiving neuron to reset and enter a refractory period. In practice we implement this by treating the reset neuron as if it has recently spiked by using the update:

$$t_{lastspike} \leftarrow t,$$

where t is the current time.

3.2 Using Spike Timing to Find the Max

Jarrett et al. [45] showed in a comparison of object recognition architectures that the top performing algorithms are those with a hierarchical structure incorporating a non-linearity, although some more recent works show similar performance with a single layer of neurons, but at the expense of increased computational complexity and training difficulty [46]. In the case of the popular HMAX [39] model, this non-linearity is a maximum operation in the pooling stages (C1 and C2). Finding this maximum requires comparing the responses of all units within the region to be pooled. This maximum value is then passed through to the next layer, irrespective of how large or small the value is. In other words, the maximum value is passed to the next layer, regardless of its value (so long as it is the maximum).

In the HFirst architecture we observe which neuron responds first, and judge that neuron to have the maximal response to the stimulus. This is based on two main observations. Firstly, that sharper edges (larger spatial gradients) result in larger temporal contrast (1), therefore generating events sooner than less sharp edges. Secondly, the higher
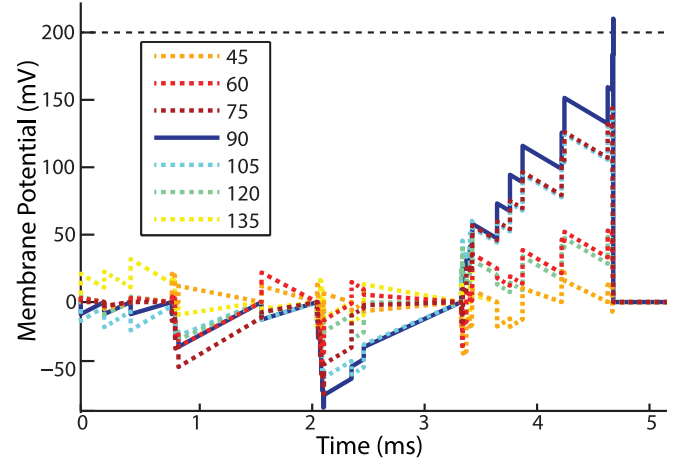

Fig. 3. Competition between neurons tuned to different orientations when presented with a visual edge oriented at 90 degrees. The neuron tuned to 90 degrees is strongest stimulated causing it to cross spiking threshold first and reset all other orientations.

the spatial correlation between a neuron's input weights and the spatial pattern of incoming spikes, the stronger it will be activated (see Fig. 3). The strongest activated neuron will cross its spiking threshold before other neurons, thereby providing an indication that its response is strongest. Using this mechanism there is no need to compare neuron responses to each other, rather we simply observe which neuron generated an output first. The first spike from a pooling region can then be used to reset other orientations through lateral reset connections, thereby ensuring that non-maximal responses are not propagated through to subsequent layers.

Fig. 3 shows how neurons tuned to different orientations will respond when an edge is presented. The neuron tuned to the orientation of the edge (90 degrees, solid line) is strongest activated and crosses the spiking threshold before other neurons (dotted lines). Neurons tuned to orientations similar to the stimulus (75 and 105 degrees) are next strongest activated, but are reset by the neuron sensitive to 90 degrees (since it spiked first). Neurons tuned to orientations below 45 degrees and above 135 degrees are not shown to reduce figure clutter.

The "time to first spike" approach simplifies computation of the max. It indicates which neuron has the strongest response, and through the time at which the spike is elicited it conveys how strong the response is. However, if no neuron was activated strongly enough to generate an output spike, no first-spike is detected and no output spikes are generated. This is an important property ensuring that no computation is performed when there is insufficient activity in the scene. Much like the front end sensor, which represents lack of stimulus (temporal contrast) through a lack of data, HFirst represents the lack of a strong enough neuron activation through a lack of output spikes.

4 ASYNCHRONOUS HFIRST ARCHITECTURE

HFirst is structured in a similar manner to hierarchical neural models [39], [43], which consist of four layers, named Simple 1 (S1), Complex 1 (C1), Simple 2 (S2), and
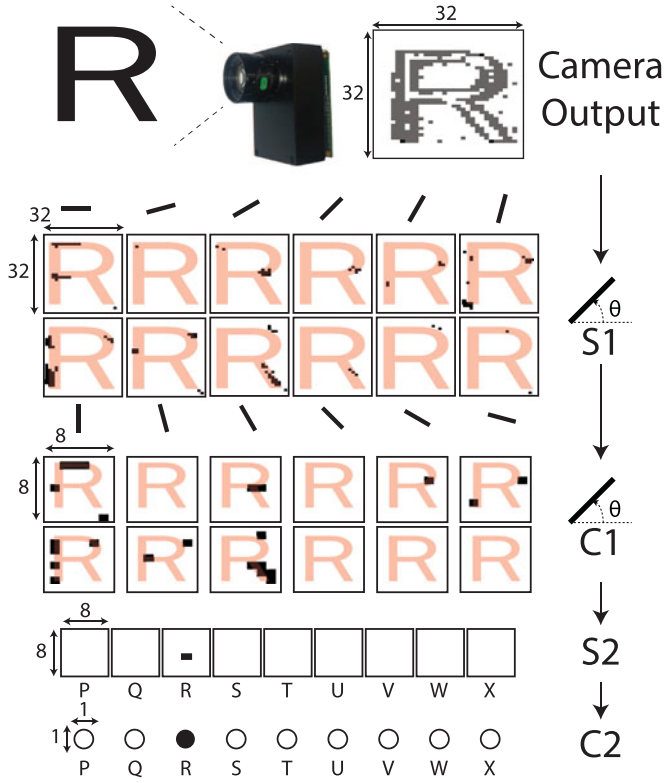
Fig. 4. The HFirst model architecture, consisting of four layers (S1, C1, S2, C2). Only a 32 × 32 pixel cropped region of real data extracted from the model is shown to ease visibility while demonstrating recognition of the character 'R'. Black dots represent data from the model. The character 'R' has been superimposed on top of the S1 and C1 data to aid explanation. The size of the (cropped) data is shown at the left of each layer (Table 1 shows the sizes for the full model). The S1 layer performs orientation extraction at a fine scale, followed by a pooling operation in C1. Note that due to lateral reset in C1, some S1 responses are blocked (for example, the last three orientations on the bottom row). The S2 layer combines responses from different orientations, but maintains spatial information. The C2 layer pools across all S2 spatial locations, providing only a single output neuron for each character.

Complex 2 (C2). In these frame base architectures, cells in simple layers densely cover the scene and respond linearly to their inputs, while cells in complex layers have a non-linear response and only sparsely cover the scene. The layers and manner in which computation is performed in HFirst differs considerably from previous implementation of similar computational models of object recognition in cortex [39], [47], [48]. The Simple layers in HFirst are in fact non-linear due to the use of a spike threshold and binary spike output. In the remainder of this section the form and function of each HFirst layer is described. The same neuron model is used for all layers, but with different parameters and connectivity. The network architecture is shown in Fig. 4, and the parameters for each stage are shown in Table 1.

4.1 Layer 1: Gabor Filters

The S1 layer densely covers the scene with even Gabor filters at 12 orientations. All filters are 7 × 7 pixels, resulting in 12 filters at each pixel. These filters are designed to pick up sharp edges. Filter kernels are generated with the same equation as in Serre et al. [39], repeated below for convenience

$$F_\theta(u, v) = e^{\left(-\frac{u_0^2 + \gamma^2 v_0^2}{2\sigma^2}\right)} \cos\left(\frac{2\pi}{\lambda} u_0\right)$$

$$u_0 = u\cos\theta + v\sin\theta$$

$$v_0 = -u\sin\theta + v\cos\theta, \tag{2}$$

where u and v are horizontal and vertical location in pixels. u_0 and v_0 are used to effect a rotation which orients the filter. Parameters of $\lambda = 5$ and $\sigma = 2.8$ were used to generate the synaptic weights. θ varies from 0 to 165 degrees in increments of 15 degrees.

S1 neurons are divided into adjacent non-overlapping 4×4 pixel regions, referred to as S1 units. Each S1 unit feeds into 12 C1 neurons, one for each orientation. C1 neurons have lateral reset connections between orientations to perform the max operation discussed in Section 3.2. C1 neurons use a very low threshold voltage to ensure that a single input spike is sufficient to generate an output spike (provided the neuron is not under refraction).

The refractory period in C1 saves computation by reducing the number of spikes which need to be routed within the architecture. Limiting the firing rate is also important to ensure that no single C1 neuron can fire rapidly enough to single handedly elicit a spike from an S2 neuron.

4.2 Layer 2: Template Matching

S2 neurons densely cover C1 neurons, with each receiving inputs from 8×8 C1 neurons of all orientations. S2 receptive fields are created during a training phase as described below.

A simple activity tracker [34] is used to track training objects and compensate for their motion to generate a static 32×32 pixel view of the object. This stabilised view is processed by S1 and C1, and the number of spikes of each orientation originating from each C1 neuron is counted. Note that due to the non-overlapping S1 units, the 32×32 pixel input region feeds into 8×8 C1 neurons, which is the size of an S2 receptive field in HFirst (see Table 1).

The counts generated in this manner constitute the synaptic weights (or input kernel) for the S2 neuron sensitive to this object. A separate neuron is required for each object to be recognized. For each neuron, synaptic weights are normalised to have an l_2 norm of 100. Finally, since negative spike counts are not possible, all zero valued weights are replaced with inhibitory values (-1) to reduce noise sensitivity. A copy of each trained neuron is then implemented at every location, allowing detection of all trained objects at all locations.

Fig. 5 shows an example of a learnt S2 receptive field for recognizing the character 'G'. The figure shows how the highest input synapse weights are assigned to locations where the orientation of character's edges match the orientation to which the underlying C1 neurons are tuned.

S2 neuron spikes reset all other S2 neurons within an 8×8 region sensitive to other classes of objects, thus implementing the max operation discussed in Section 3.2. Furthermore, by only resetting neurons sensitive to *other* object classes, the detected object class is given a "head start" in the race to first spike in the nearby region. This can be seen as using the detection to create a prior expectation of detecting that object again nearby.
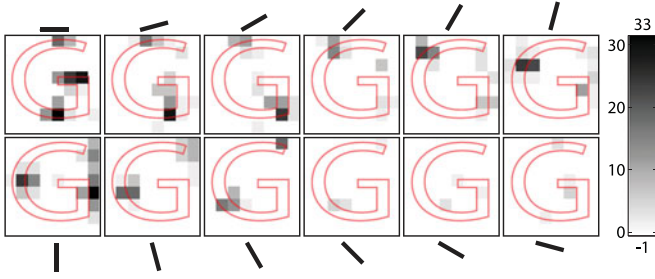
Fig. 5. The receptive field of an S2 neuron trained to recognize the character 'G'. The neuron receives inputs from an 8 × 8 (x×y) C1 region and from all 12 orientations (orientations are indicated by the oriented blacked bars). Dark regions indicate strong excitatory weights and can be seen to fall along edges of the character wherever edge orientation matches the C1 neuron orientation. Weaker response between 135 and 165 degrees (bottom right) are due to the direction of motion of the character during training (roughly 150 degrees). Motion perpendicular to the direction of motion is required to elicit temporal contrast, as shown in (1) and discussed in Section 2. After normalization the weights in this example range from −1 to 33 mV, indicated by the bar on the right.

An optional C2 layer can be used to pool all responses from all S2 locations for classification. The C2 layer is not always used because it discards information regarding the location of the object, which can be particularly useful when multiple objects of interest are simultaneously present in the scene.

4.3 Classifier

A basic classifier outputs the soft probabilities for the object belonging to each class. The probability $P(i)$ of an object belonging to class i is calculated as

$$P(i) = \frac{n_i}{\sum_i n_i}, \tag{3}$$

where n_i is the number of spikes elicited by S2 neurons sensitive to the ith class. When $\sum_i n_i = 0$ we assign $P(i) = 0$ for all classes.

If we wish to force the classifier to choose only a single class, we can assign the output class y as

$$y = \max_i(n_i). \tag{4}$$

We have no neuron to respond to lack of an object in a scene. Lack of an object results in lack of positive detections. This is a fundamental concept of the computing and sensing paradigm we use. Lack of information is not communicated, but is rather represented by a lack of communicated data.

5 IMPLEMENTATION

In this section we briefly analyse computational requirements. The number of input spikes generated by the front end sensor varies with scene activity and dictates the required computation since neuron updates are only performed when spikes are received. We analyse computation as a function of the number of input and output spikes for each layer. A worst case scenario is used which assumes that a neuron is updated every time it receives a spike (ignoring the refractory period).

A parallelised and pipelined FPGA implementation was programmed to run in real-time on the Opal Kelly XEM6010-LX150 board, which includes a Xilinx Spartan 6 XC6SLX150-2 FPGA. The model operates on a 128 × 128 pixel input. The implementation runs at a clock frequency of 100MHz and uses internal block RAM without relying on external RAM. The final output of the system consists of S2 output spikes, although access is also provided to spikes from intermediate layers for characterization.

5.1 S1 and C1: Gabor Filters

Each input spike in S1 routes to all S1 neurons within a 7×7 pixel region. There are 12 S1 neurons at each pixel location (one per orientation), resulting in 12 × 7 × 7 = 588 synapse activations per input spike.

For FPGA implementation, 84 synapses update in parallel, requiring seven clock cycles to update all 588 synapses, allowing the S1 stage to sustain throughput of 14 M events per second (eps).

Each S1 output spike excites a single C1 neuron, and resets the 11 C1 neurons sensitive to other orientations, resulting in 12 C1 synapse activations per S1 output spike. C1 updates all 12 synapses in parallel and can process 25 M input events per second.

5.2 S2 and C2: Template Matching

Each input spike to S2 routes to all S2 neurons within an 8 × 8 region. If N_y denotes the number of classes to be classified, then there will be N_y neurons at each location, and each input spike will activate $N_y \times 8 \times 8 = 64 N_y$ input synapses. Each S2 output spike resets all S2 neurons lying within an 8 × 8 region around where the spike originated. So, for every S2 output spike $N_y \times 8 \times 8 = 64 N_y$ S2 lateral reset synapses are activated.

The FPGA implementation of S2 can update N_y neurons in parallel, requiring 64 clock cycles to process each input or output spike. The number of C2 input synapses activated is equal to the number of S2 neuron output spikes. The C2 stage is optional and not implemented in FPGA.

In HFirst there are no zero valued synapses in S2. Synapses which are not activated during training are assigned an inhibitory synaptic weight of −1 (see Section 4.2). The number of synapses in S2 could be significantly reduced by instead assigning a weight of 0 to these synapses and optimizing them out of the model. However, such an optimization would introduce significant additional complexity in pipelining for the FPGA implementation. The FPGA implementation benefits far more from the simplified pipelining which results from having a dense regular connection structure where all synapses are implemented. This connection structure is also more general, allowing the synaptic weights to be easily reprogrammed.

The regular connection structure also saves memory by ensuring that when a neuron is updated, all co-located neurons will also be updated. Updating all co-located neurons simultaneously allows us to store only a single time value to indicate when all neurons at that location were updated, rather than storing a separate time value to indicate when each individual neuron was last updated (Section 3.1 shows how the time value is used in the neuron update). This memory saving is important because memory availability is

TABLE 2
Required Computation and Resources

Stage	S1	C1	S2	C2
Synapse updates per event	588	12	$64 N_y$	1
Throughput events/sec	14M	25M	1M	100M
DSP blocks	16	0	1	0
Block RAM	128	2	N_y+1	0

the limiting factor in scaling the model to higher resolution, as shown in the next section.

5.3 Scaling to Higher Resolution

When scaling to higher resolutions two main factors need to be considered: memory requirements, and computational requirements. Required memory scales linearly with the number of neurons in the model, which in turn scales linearly with the number of input pixels. Computational requirements scale linearly with the input event rate.

With 36 classes ($N_y = 36$), 167 Block RAMs are used for HFirst (see Table 2), plus an additional 10 for pipeline FIFOs and USB IO, resulting in a total of 177 of the available 268 Block RAMs being used for 128×128 pixel input resolution.

Digital signal processing (DSP) blocks are blocks within the FPGA containing dedicated hardware for performing multiplication and addition. The number of multiplications which can be performed per second is a limiting factor in many algorithms, particularly for visual processing algorithms which compute kernel responses using convolution. Optimization of these algorithms typically involves optimizing memory access and pipelining to maximise utilization of hardware multipliers (see [49] for an example). High end GPUs and FPGAs contain thousands of hardware multipliers.

In HFirst only 17 of our FPGA's 180 DSP blocks are used and these 17 DSP blocks are only utilized a small percentage of the time due to the temporal sparsity of the AER data. For HFirst, internal FPGA memory is the limiting resource when increasing resolution. Internal memory requirements scale with the input sensor resolution, while the number of DSP blocks required will scale with maximum sustained input event rate the model is required to handle.

The current FPGA implementation can handle a sustained 14M events per second input event rate, while bursts of up to 100M eps (limited by FPGA clock speed of 100 MHz) can be handled for durations up to 5 μs (limited by FIFO buffer depth). Larger FIFO buffers can be used, but are unnecessary. At 128×128 resolution, event rates for typical scenes are around 1Meps. The latest ATIS can generate events at a peak rate of 25 M eps, and sustain a maximum rate of 15M eps at 304×240 pixel resolution. 14Meps is therefore a very high rate for 128×128 pixel resolution. Using additional DSP blocks, the maximum sustainable event rate can be increased by 1M eps per block used.

5.4 Power Consumption

The FPGA board on which HFirst was implemented also performs other tasks in parallel as part of normal operation of the ATIS sensor (powering and controlling the ATIS, as well as interfacing to a host PC). Implementing HFirst in

Fig. 6. The test setup used to acquire the character dataset, consisting of a motorised rotating barrel covered with printed letters viewed by a DVS [21].

addition to the other tasks on the FPGA increases power consumption by 150mW for static scenes (little to no processing happening), and by a further 100 mW for the highest activity scene we could generate. We therefore estimate HFirst power consumption to be between 150 and 250 mW depending on scene activity. These measurements are done at the board's power supply and include losses due to inefficiencies in the onboard switching regulators.

6 TESTING

HFirst was tested on two tasks. The first consists of recognizing pips on poker cards as they are shuffled in front of the sensor. The poker card task has been previously tackled [35] and was chosen to provide a direct comparison with previously published works. The second task is a simulated reading task in which characters are recognized as they move across the field of view using the test setup shown in Fig. 6. Examples of recordings used for each task are shown in Fig. 7. For both tasks, HFirst was implemented in Matlab simulation, coupled with a reconfigurable C++ function for increased speed.

6.1 Poker Cards

For the poker card task data was provided by Linares-Barranco [35] who captured the data using the sensitive DVS sensor [22]. The dataset consists of 10 examples for

Fig. 7. Examples of the stabilised characters and cards pip views used for training. Each example measures 32×32 pixels and shows 1.7 ms of data.

each of the four card types (spades, hearts, diamonds, and clubs). For each of 10 different trials, non-overlapping test and training sets were chosen such that each contained five examples of each pip. For each pip in the training set, all five examples were concatenated into a single sequence from which the S2 layer kernel was generated. To provide a close comparison with the previously published task, we also tested on the stabilised and extracted pips.

Additional tests were performed in which lateral reset connections were removed from the model to investigate the value of the timing approach to computing the max. Finally, the advantage of having orientation extraction and pooling in S1 and C1 were investigated by bypassing these stages.

6.2 Character Recognition

Thirty six characters (0-9 and A-Z) were printed on the surface of a barrel which was rotated at 40 rpm while viewed by the DVS [21] as shown in Fig. 6. Data was recorded over two full rotations of the barrel, thereby providing two recordings for each character. For each of 10 trials, non-overlapping test and training sets were randomly chosen such that every character appears once in each set. Training and testing was then performed using an automated script.

Training of the second layer of HFirst is performed on a stabilised view of a moving object, and therefore requires knowledge of the object location, which is acquired through tracking. However, for testing we use moving sequences instead of stabilised views, removing the need for tracking.

As with the card task, The character recognition task was also used to investigate the advantages of using reset connections for max computation, and of performing orientation extraction and pooling in S1 and C1 respectively.

Further testing was performed on the characters to show that HFirst can detect multiple objects simultaneously present in the scene, and to investigate the impact of timing jitter introduced during training and testing.

Finally the importance of precise timing was investigated by artificially altering spike times in the recordings and observing the effect on HFirst accuracy.

7 RESULTS

Results from testing are summarised in Table 3, and discussed in the sections below. The S1 and C1 columns show the total number of activated synapses in each of these layers. For S2, the S2 and $S2_{rst}$ columns show the number of activated feedforward (from C1) and lateral reset synapses respectively.

7.1 Cards

HFirst classified the stabilised and extracted card pips with an accuracy of 97.5% ± 3.5% using an S2 threshold of 150 mV. Chance for this task is 25 percent. The average duration of a test example was 23 ms, and consisted of 4.3 k input spikes, which elicited 73 C1, and 2.8 S2 spikes. The S1/C1 and S2/C2 layers took on average 102 and 0.7 ms respectively per example to simulate in Matlab using a single thread on an Intel Xeon X5675 processor running at 3.07 GHz. The FPGA implementation

TABLE 3
Detection Accuracy and Required Computation

Task	Accuracy	Input Synapse Activations			
	%	S1	C1	S2	$S2_{rst}$
HFirst Cards					
Full model	97.5 ± 3.5	2.6M	10k	19k	710
No S1, C1 reset	51.6 ± 4.4	2.6M	3.8k	127k	79k
No S2, C2 reset	72.3 ± 3.8	2.6M	10k	19k	-
No reset	24.9 ± 0.1	2.6M	3.8k	127k	-
Bypass S1	49.1 ± 5.4	-	4.3k	37k	20k
Bypass S1 and C1	60.7 ± 3.5	-	-	1.1M	333k
CNN Cards					
Spiking [35]	91.6	-	-	-	-
Frame based [35]	95.2	-	-	-	-
HFirst Characters					
Full model	84.9 ± 1.9	8.4M	40k	720k	159k
No S1, C1 reset	70.4 ± 5.8	8.4M	8.3k	2.4M	309k
No S2, C2 reset	56.7 ± 0.9	8.4M	40k	720k	-
No reset	4.6 ± 0.1	8.4M	8.3k	2.4M	-
Bypass S1	31.2 ± 4.1	-	14k	1.6M	1.1M
Bypass S1 and C1	81.4 ± 3.8	-	-	33M	32M

simulates the network in real-time, with latency ≤ 2 μs in response to incoming events.

Removing lateral reset in the first layer decreases recognition accuracy to 51.6% ± 4.4%, while removing lateral reset connections in the second layer decreases recognition accuracy to 72.3% ± 3.8%, and removing lateral reset connections in both layers reduces recognition accuracy to chance levels, while increasing the average number of spikes elicited to 309 and 66 for C1 and S2 respectively. These results suggest that using the first spike mechanism improves performance, both in terms of computational efficiency, and in terms of recognition accuracy.

For the card classification task which only has four output classes, bypassing the first layers reduces the required computation at the cost of recognition accuracy.

7.2 Characters

HFirst classified the moving letters with an accuracy of 84.9% ± 1.9% using an S2 threshold of 200 mV. Chance for this task is 2.8 percent. The average duration of a test example was 112 ms, and consisted of 14k input spikes, which elicited 313 C1, and 27 S2 spikes on average. The S1/C1 and S2/C2 layers took on average 365 and 28 ms respectively per example to simulate in Matlab using a single thread on an Intel Xeon X5675 processor running at 3.07 GHz. As with the card pip task, the FPGA implementation easily runs in real-time with latency ≤ 2 μs.

Next we investigated the effects of bypassing the first layers of HFirst and performing template matching directly on the input events. This modification resulted in an accuracy of 81.4% ± 3.8%, which is not too different from the performance of the full model. However, bypassing the S1 and C1 layers also increases the required computation significantly, suggesting that performing orientation extraction and pooling in S1 and C1 is actually more computationally efficient. The same is not true for the cards task where only four classes are present, but is true whenever 10 or more output classes are required. This increased computational
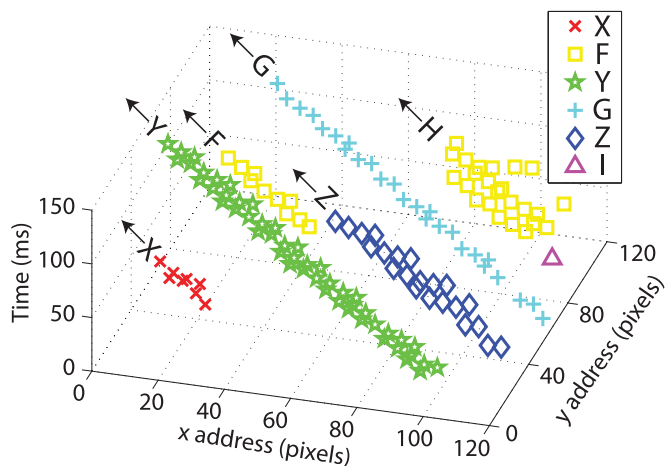
Fig. 8. HFirst S2 layer spikes (indicated by markers) over a 150 ms time period in response to the character data. This figure shows the ability of HFirst to detect multiple characters in the scene simultaneously. Both location of the objects and their class are indicated by S2 spikes. The 'X', 'F', 'Y', and 'G' characters are correctly detected, but the character 'H' is misclassified, being mistaken for an 'I' or 'F' at different times.
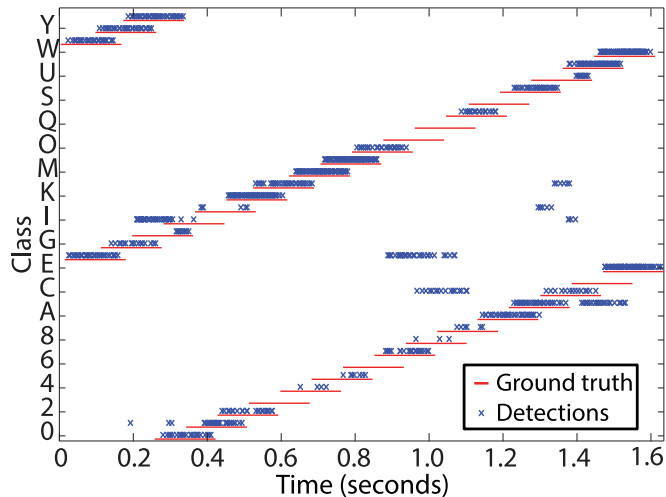

Fig. 9. Detection of characters for a single rotation of the barrel. Only every second character is labelled on the vertical axis to reduce clutter. Red lines indicate when each character is present in the visual field, while blue crosses mark detections made by HFirst. Note that up to four characters are present in the scene at any one time.

requirement is also obvious when observing the time taken for simulation, which increased by 50 fold to an average of 19.7 seconds per example in Matlab.

7.3 Detecting Multiple Objects Simultaneously

After testing the model performance on individual characters, we verified that it can detect multiple characters simultaneously present in the scene. Fig. 8 shows 150 ms worth of S2 outputs with multiple characters simultaneously visible in the scene. The S2 responses indicate both the object class and location. In this example the letters 'X', 'F', 'Y', and 'G' are all accurately detected as they pass across the scene. Later, the letters 'Z' and 'H' enter the scene. The 'Z' is accurately detected, but the 'H' is erroneously detected as an 'F' and 'I' at different points in time. The 1 in 6 error for these characters is in agreement with the 84.9% ± 1.9% accuracy reported overall.

Fig. 9 shows output detections for a single full rotation of the barrel, comparing the times at which letters were detected (or missed) to the ground truth of when they were present in the scene.

7.4 Effect of Timing Jitter

In the front end AER sensor, the latency of pixel responses and of the AER readout can vary, resulting in timing jitter in the spikes feeding into S1. All of our tests are performed on real recordings and therefore include some jitter. In order to investigate the effect of increased timing jitter on the model, we artificially added additional jitter to the recordings used for training and testing. Jitter times for each spike were randomly chosen from a Gaussian distribution and the effect of varying the standard deviation of the distribution is shown in Fig. 10. Changing the mean of the Gaussian distribution adds a constant time offset to all spikes and has no effect on accuracy. The accuracy for each standard deviation value is again obtained as the mean of 10 random test and training splits performed on the character database. Two tests were run, in the first test additional jitter was introduced in the training data (Fig. 10a) and the test data

was left unaltered. In the second test (Fig. 10b) the training data was left unaltered and additional jitter was introduced only in the test data.

Training is performed on tracked and stabilized views of the characters, thus for the purposes of training, the characters appears static. HFirst can therefore tolerate high timing jitter because even when a spike's time is changed, it will still occur in the correct location relative to the center of the character. Accuracy drops off significantly only when the standard deviation of the jitter exceeds 100 ms, which is comparable to the length of the recording itself (112 ms).
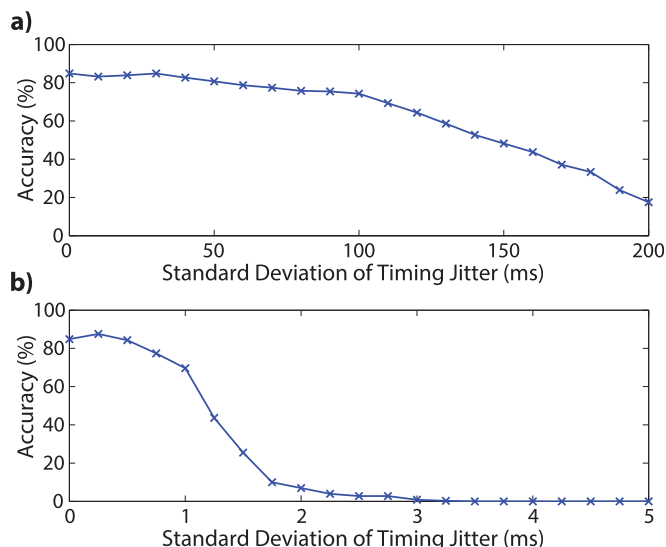

Fig. 10. The effect of timing noise on recognition accuracy for the character recognition task. Adding Gaussian noise to the stabilized training data (a) has little effect on accuracy because even when delayed, spikes occur in the correct location relative to the character center. Accuracy drops off significantly only when the timing jitter is large enough to cause the training data spikes to be too spread in time. Adding even a small degree of Gaussian noise to the moving characters used for testing (b) causes accuracy to drop off significantly because by the time the delayed (jittered) spikes arrive at the S1 inputs, the character has already moved on to a new location.

Recognition is performed on moving views of the characters which are crossing the field of view at roughly 1 pixel/ms. Delaying a spike by even a few milliseconds (Fig. 10b) will cause the spike to occur in the wrong location relative to the center of the character (because the character center will have moved during the delay period). Therefore, even a few milliseconds of timing jitter will cause a significant decrease in recognition accuracy.

8 DISCUSSION

In this paper we have described a spiking neural network for visual recognition dubbed "HFirst". HFirst exploits timing information in the incoming visual events to implement a time-to-first spike operation as a temporal Winner-Take-All operation with lateral reset to block responses from other neurons in the same pooling area. Computationally, this temporal WTA is significantly simpler than the MAX operation typically used in hierarchical models.

HFirst operates on change detection data from AER sensors. Each pixel in these sensors adapts individually to ambient lighting conditions, which to a large extent removes dependence on lighting conditions. This removes the need for normalization of oriented Gabor responses in HFirst, which is another computationally intensive task (division) required by the standard HMAX model and other CNN implementations.

Thus far HFirst has been tested on simple objects, and neurons in the second layer of HFirst directly detect the presence of these objects, allowing HFirst to simultaneously detect multiple objects in the scene, which is not typically possible with CNNs.

Masquelier and Thorpe [43] used STDP to learn more complex features, and a powerful radial basis function (RBF) classifier which allows recognition of more complex objects (motorcycles and faces from Caltech 101). Their approach used STDP to extract features with high correlation between training examples, even though these features appear at different locations. This removes the need to precisely track and stabilise a view of an object for training. However, the model only operates on static images, removing the problem of moving stimuli, and objects are already centered in the Caltech 101 database (although features do not always appear at the same location). A second major difference is that HFirst operates continuously, whereas Masquelier et al. present images to their model sequentially, requiring the system to be reset before each image presentation.

In a recent PAMI paper, Perez-Carrasco et al. [35] reported an accuracy ranging from 90.1 to 91.6 percent for the card pip task using a five layer spiking CNN. They kindly provided us with their data and for the same task we report accuracy of 97.5% ± 3.5%. However, we compute accuracy differently to Perez-Carrasco et al. Their CNN implementation includes separate "positive" and "negative" responses to represent the presence or absence for each object, and both these responses are used in their calculation of accuracy. HFirst has no "negative" responses, which prevents us from using the same equation. Instead, HFirst provides only positive responses, and does not respond when no objects of interest are present in the scene. Nevertheless, if we consider a lack of response from a neuron to be a "negative" response, then we can use the same equation. Doing so marginally increases our accuracy to 98.8% ± 1.9% because correct "negative" responses are rewarded, even when "positive" responses are incorrect.

The card pip task was also used to investigate the benefits of including lateral reset, by showing that removal of lateral reset connections in the first, second, or both layers consistently reduces recognition accuracy, while simultaneously increasing computational requirements.

Given the high accuracy of the full HFirst model on the card pip recognition task, a second more difficult character recognition task was constructed and was also used to investigate the benefits of a multi-layer model. Bypassing the first layer decreased accuracy from 84.9% ± 1.9% to 81.4% ± 3.8%, suggesting that the first layer increases recognition accuracy. Perhaps more importantly, the first layer significantly reduces computational requirements for the character recognition task. The same was not true for the card recognition task because it consists of very few classes (4), but as the number of classes increases, so does the number of neurons in S2, therefore making it more important to have the S1 and C1 layer to reduce the number of spikes reaching S2.

The leaky integrate and fire neurons used in HFirst essentially perform coincidence detection on input spikes arriving in a specific spatial pattern. A neuron will only generate an output spike if enough input spikes matching this pattern are received within a sufficiently short time period. Under ideal circumstances (no noise), the projection of an object moving between two points on the focal plane will generate the same number of spikes from the AER sensor, regardless of the speed of the object. However, the speed of the object will determine the time period over which these spikes are generated, with slow moving objects not generating spikes at a high enough rate to elicit a response from HFirst layer 1 neurons, but this can be overcome through active sensing, by using a small motion or vibration of the sensor to elicit an egomotion induced velocity on the image plane.

9 CONCLUSION

We have presented an HMAX inspired hierarchical SNN architecture for visual object recognition dubbed 'HFirst'. The architecture uses an SNN to exploit the precise spike timing provided by asynchronous change detection vision sensors to simplify implementation of a non-linear pooling operation commonly used in bio-inspired recognition models. HFirst obtains the best reported accuracy on a card pip recognition test and results for a second, far more difficult character recognition task have also been presented. The low computational requirements of the HFirst model allow for real time implementation on an Opal Kelly XEM6010 FPGA board which interfaces directly with the vision sensor, and is both narrower and shorter than a credit card in size.

REFERENCES

[1] T. J. Gawne, T. W. Kjaer, and B. J. Richmond, "Latency: Another potential code for feature binding in striate cortex," *J. Neurophysiol.*, vol. 76, no. 2, pp. 1356–1360, 1996.

[2] M. Greschner, A. Thiel, J. Kretzberg, and J. Ammermller, "Complex spike-event pattern of transient on-off retinal ganglion cells," *J. Neurophysiol.*, vol. 96, no. 6, pp. 2845–2856, 2006.

[3] T. Masters, *Practical Neural Network Recipes in C++*. San Mateo, CA, USA: Morgan Kaufmann, 1993.

[4] M. Egmont-Petersen, D. de Ridder, and H. Handels, "Image processing with neural networks: A review," *Pattern Recognit.*, vol. 35, no. 10, pp. 2279–2301, 2002.

[5] C. M. Bishop, *Neural Networks for Pattern Recognition*, London, U.K.: Oxford Univ. Press, 1995.

[6] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello, "Hardware accelerated convolutional neural networks for synthetic vision systems," in *Proc. IEEE Int. Symp. Circuits Syst.*, Jun. 2010, pp. 257–260.

[7] B. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proc. IEEE*, vol. 102, no. 5, pp. 699–716, May 2014.

[8] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.

[9] D. H. Goldberg, G. Cauwenberghs, and A. G. Andreou, "Probabilistic synaptic weighting in a reconfigurable network of VLSI integrate-and-fire neurons," *Neural Netw.*, vol. 14, no. 67, pp. 781–793, 2001.

[10] J. Park, T. Yu, C. Maier, S. Joshi, and G. Cauwenberghs, "Live demonstration: Hierarchical address-event routing architecture for reconfigurable large scale neuromorphic systems," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2012, pp. 707–711.

[11] J. Schemmel, A. Grubl, S. Hartmann, A. Kononov, C. Mayr, K. Meier, J. Millner, J. Partzsch, S. Schiefer, S. Scholze, R. Schuffny, and M. Schwartz, "Live demonstration: A scaled-down version of the brainscales wafer-scale neuromorphic system," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2012, p. 702.

[12] E. Painkras, L. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D. Lester, A. Brown, and S. Furber, "SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation," *IEEE J. Solid-State Circuits*, vol. 48, no. 8, pp. 1943–1953, Aug. 2013.

[13] C. Eliasmith and C. H. Anderson, *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. Cambridge, MA, USA: MIT Press, 2004.

[14] D. F. M. Goodman and R. Brette, "Brian: A simulator for spiking neural networks in python," *Frontiers Neuroinform.*, vol. 2, no. 5, 2008.

[15] A. P. Davison, D. Brüderle, J. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger, "Pynn: A common interface for neuronal network simulators," *Frontiers Neuroinform.*, vol. 2, no. 11, 2009.

[16] H. Markram, K. Meier, T. Lippert, S. Grillner, R. Frackowiak, S. Dehaene, A. Knoll, H. Sompolinsky, K. Verstreken, J. DeFelipe, S. Grantk, J.-P. Changeuxl, and A. Sariam, "Introducing the human brain project," *Procedia Comput. Sci.*, vol. 7, pp. 39–42, 2011.

[17] A. P. Alivisatos, M. Chun, G. M. Church, R. J. Greenspan, M. L. Roukes, and R. Yuste, "The brain activity map project and the challenge of functional connectomics," *Neuron*, vol. 74, no. 6, pp. 970–974, 2012.

[18] E. Culurciello, R. Etienne-Cummings, and K. Boahen, "A biomorphic digital image sensor," *IEEE J. Solid-State Circuits*, vol. 38, no. 2, pp. 281–294, Feb. 2003.

[19] K. A. Zaghloul and K. Boahen, "Optic nerve signals in a neuromorphic chip i: Outer and inner retina models," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 4, pp. 657–666, Apr. 2004.

[20] M. Mahowald, *An Analog VLSI System for Stereoscopic Vision*, series Kluwer International Series in Engineering and Computer Science. Norwell, MA, USA: Kluwer, 1994.

[21] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 $\times$ 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008.

[22] T. Serrano-Gotarredona and B. Linares-Barranco, "A 128 $\times$ 128 1.5% contrast sensitivity 0.9% FPN 3 μs latency 4 mw asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers," *IEEE J. Solid-State Circuits*, vol. 48, no. 3, pp. 827–838, Mar. 2013.

[23] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 db dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, Jan. 2011.

[24] T. Delbruck, B. Linares-Barranco, E. Culurciello, and C. Posch, "Activity-driven, event-based vision sensors," in *Proc. IEEE Int. Symp. Circuits Syst.*, Jun. 2010, pp. 2426–2429.

[25] F. Folowosele, R. J. Vogelstein, and R. Etienne-Cummings, "Real-time silicon implementation of v1 in hierarchical visual information processing," in *Proc. Biomed. Circuits Syst. Conf.*, 2008, pp. 181–184.

[26] R. J. Vogelstein, U. Mallik, E. Culurciello, G. Cauwenberghs, and R. Etienne-Cummings, "Saliency-driven image acuity modulation on a reconfigurable silicon array of spiking neurons," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 1457–1464.

[27] R. J. Vogelstein, U. Mallik, E. Culurciello, G. Cauwenberghs, and R. Etienne-Cummings, "A multichip neuromorphic system for spike-based visual information processing," *Neural Comput.*, vol. 19, no. 9, pp. 2281–2300, 2007.

[28] R. J. Vogelstein, U. Mallik, J. T. Vogelstein, and G. Cauwenberghs, "Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 253–265, Jan. 2007.

[29] S. Schraml and A. Belbachir, "A spatio-temporal clustering method using real-time motion analysis on event-based 3D vision," in *Proc. IEEE Comput. Vis. Pattern Recognit. Workshops*, 2010, pp. 57–63.

[30] P. Rogister, R. Benosman, S.-H. Ieng, P. Lichtsteiner, and T. Delbruck, "Asynchronous event-based binocular stereo matching," *IEEE Trans. Neural Netw.*, vol. 23, no. 2, pp. 347–353, Feb. 2012.

[31] R. Benosman, P. Rogister, C. Posch, L. Sio-Hoï, "Asynchronous event-based Hebbian epipolar geometry," *IEEE Trans. Neural Netw.*, vol. 22, no. 11, pp. 1723–1734, Nov. 2011.

[32] R. Benosman, S.-H. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan, "Asynchronous frameless event-based optical flow," *Neural Netw.*, vol. 27, pp. 32–37, 2012.

[33] G. Orchard and R. Etienne-cummings, "Bio-inspired visual motion estimation," *Proc. IEEE*, vol. 102, no. 10, pp. 1520–1536, Oct. 2014.

[34] T. Delbruck and P. Lichtsteiner, "Fast sensory motor control based on event-based hybrid neuromorphic-procedural system," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 845–848.

[35] J. Perez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, and B. Linares-Barranco, "Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward convnets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2706–2719, Nov. 2013.

[36] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, L. Camuñas-Mesa, R. Berner, M. Rivas-Pérez, T. Delbruck, L. Shih-Chii, R. Douglas, P. Hafliger, G. Jimenez-Moreno, A. C. Ballcels, T. Serrano-Gotarredona, A. J. Acosta-Jimenez, and B. Linares-Barranco, "CAVIAR: A 45k neuron, 5M synapse, 12G connects/s AER hardware sensory–processing–learning–actuating system for high-speed visual object recognition and tracking," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1417–1438, Sep. 2009.

[37] S. Chen, P. Akselrod, B. Zhao, J. Perez-Carrasco, B. Linares-Barranco, and E. Culurciello, "Efficient feedforward categorization of objects and human postures with address-event image sensors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 2, pp. 302–314, Feb. 2012.

[38] S.-C. Liu and T. Delbruck, "Neuromorphic sensory systems," *Current Opinion Neurobiol.*, vol. 20, no. 3, pp. 288–295, 2010.

[39] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, "Robust object recognition with cortex-like mechanisms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 3, pp. 411–426, Mar. 2007.

[40] T. Serre, "Learning a dictionary of shape-components in visual cortex: Comparison with neurons, humans and machines," Ph.D. dissertation, Dept. Brain Cognitive Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, 2006.

[41] F. Folowosele, R. J. Vogelstein, and R. Etienne-Cummings, "Towards a cortical prosthesis: Implementing a spike-based HMAX model of visual object recognition in silico," *IEEE J. Emerging Sel. Topics Circuits Syst.*, vol. 1, no. 4, pp. 516–525, Dec. 2011.

[42] C. Shoushun, B. Martini, and E. Culurciello, "A bio-inspired eventbased size and position invariant human posture recognition algorithm," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2009, pp. 775–778.

[43] T. Masquelier and S. J. Thorpe, "Unsupervised learning of visual features through spike timing dependent plasticity," *PLoS Comput. Biol.*, vol. 3, no. 2, p. 31, 2007.

[44] E. M. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1063–1070, Sep. 2004.

[45] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. 12th IEEE Int. Conf. Comput. Vis.*, Oct. 2009, pp. 2146–2153.

[46] O. Delalleau and Y. Bengio, "Shallow vs. deep sum-product networks," *Advances Neural Info. Processing Syst.*, vol. 24, pp. 666–674, 2011.

[47] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature Neurosci.*, vol. 2, no. 11, pp. 1019–1025, 1999.

[48] T. Serre, A. Oliva, and T. Poggio, "A feedforward architecture accounts for rapid categorization," *Proc. Nat. Acad. Sci. USA*, vol. 104, no. 15, pp. 6424–6429, 2007.

[49] G. Orchard, J. G. Martin, R. J. Vogelstein, and R. Etienne-Cummings, "Fast neuromimetic object recognition using FPGA outperforms GPU implementations," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 24, no. 8, pp. 1239–1252, Aug. 2013.

Garrick Orchard received the BSc degree in electrical engineering from the University of Cape Town, South Africa, in 2006, and the MSE and PhD degrees in electrical and computer engineering from Johns Hopkins University, Baltimore, in 2009 and 2012, respectively. He has been named a Paul V. Renoff fellow in 2007, and a Virginia and Edward M. Wysocki, Sr. fellow in 2011. He has received the JHUAPL Hart Prize for Best R&D Project in 2009, and the IEEE BioCAS best live demo award in 2012. He is currently a postdoctoral research fellow at the Singapore Institute for Neurotechnology (SINAPSE), National University of Singapore where his research focuses on developing neuromorphic vision sensors and algorithms for high speed vision tasks. His other research interests include mixed-signal VLSI, compressive sensing, navigation, and legged locomotion.

Cedric Meyer received the BSc degree in electrical engineering from the Ecole Nationale Supérieure de Cachan, France, in 2010, and the PhD degree in robotics from the University Pierre and Marie Curie, Paris, France, in 2013. His current research interests include efficient vision perception and computation for mobile robotics. He is also interested in understanding how biological visual systems encode and process visual information to perform object recognition.

Ralph Etienne-Cummings received the BSc degree in physics from Lincoln University, Pennsylvania, in 1988 and the MSEE and PhD degrees in electrical engineering from the University of Pennsylvania, in 1991 and 1994, respectively. He is currently a professor of electrical and computer engineering, and computer science at Johns Hopkins University (JHU). He is the former director of computer engineering at JHU and the Institute of Neuromorphic Engineering. He is also the associate director for Education and Outreach of the US National Science Foundation (NSF) sponsored Engineering Research Centers on Computer Integrated Surgical Systems and Technology at JHU. He was the Chairman of the IEEE Circuits and Systems (CAS) Technical Committee on Sensory Systems and on Neural Systems and Application. He was also the general chair of the IEEE BioCAS 2008 Conference. He was also a member of Imagers, MEMS, Medical and Displays Technical Committee of the ISSCC Conference from 1999-2006. He received the NSFs Career and Office of Naval Research Young Investigator Program Awards. In 2006, he was named a visiting African fellow and a Fulbright fellowship Grantee for his sabbatical at the University of Cape Town, South Africa. He was invited to be a lecturer at the National Academies of Science Kavli Frontiers Program, in 2007. He has won publication awards including the 2003 Best Paper Award of the *EURASIP Journal of Applied Signal Processing* and Best PhD in a Nutshell at the IEEE BioCAS 2008 Conference, and has been recognized for his activities in promoting the participation of women and minorities in science, technology, engineering and mathematics. His research interest includes mixed signal VLSI systems, computational sensors, computer vision, neuromorphic engineering, smart structures, mobile robotics, legged locomotion, and neuroprosthetic devices.

Christoph Posch received the MSc and PhD degrees in electrical engineering and experimental physics from the Vienna University of Technology, Vienna, Austria, in 1995 and 1999, respectively. From 1996 to 1999, he worked on analog CMOS and BiCMOS IC design for particle detector readout and control at CERN, the European Laboratory for Particle Physics in Geneva, Switzerland. From 1999 onwards he was with Boston University, Boston, MA, engaging in applied research and analog/mixed-signal integrated circuit design for high-energy physics instrumentation. In 2004, he joined the newly founded Neuroinformatics and Smart Sensors Group at AIT Austrian Institute of Technology (formerly Austrian Research Centers ARC) in Vienna, Austria, where he was promoted to principal scientist in 2007. Since 2012, he has been co-directing the Neuromorphic Vision and Natural Computation group at the Institut de la Vision in Paris, France, and has been appointed associate research professor at Universite Pierre et Marie Curie, Paris 6. His current research interests include biomedical electronics, neuromorphic analog VLSI, CMOS image and vision sensors, and biology-inspired signal processing. He has received and co-received several scientific awards including the Jan van Vessem Award for Outstanding European Paper at the IEEE International Solid-State Circuits Conference (ISSCC) in 2006, the Best Paper Award at ICECS 2007, and Best Live Demonstration Awards at ISCAS 2010 and BioCAS 2011. He has authored more than 80 scientific publications and holds several patents in the area of artificial vision and image sensing. He is a senior member of the IEEE and member of the Sensory Systems and the Neural Systems and Applications Technical Committees of the IEEE Circuits and Systems Society.

Nitish Thakor (S'78-M'81-SM'89-F'97) is a professor of biomedical engineering, electrical and computer engineering, and neurology at Johns Hopkins University, Baltimore, MD, and directs the Laboratory for Neuroengineering. He has been appointed as the provost professor, National University of Singapore, and now leads the SiNAPSE Institute, focused on neurotechnology research and development. His technical expertise is in the areas of neural diagnostic instrumentation, neural microsystem, neural signal processing, optical imaging of the nervous system, rehabilitation, neural control of prosthesis, and brain machine interface. He is the director of a Neuroengineering Training program funded by the National Institutes of Health. He has authored 250 refereed journal papers, generated 11 patents, cofounded four companies, and carries out research funded mainly by the NIH, NSF and DARPA. He was the editor-in-chief of the *IEEE Transactions on Neural and Rehabilitation Engineering* (2005-2011) and is currently the editor-in-chief of *Medical and Biological Engineering and Computing Journal*. He received the Research Career Development Award from the National Institutes of Health and a Presidential Young Investigator Award from the National Science Foundation. He is a fellow of the IEEE, the American Institute of Medical and Biological Engineering, International Federation of Medical and Biological Engineering, and founding fellow of the Biomedical Engineering Society, Technical Achievement Award from IEEE and Distinguished Alumnus award from the Indian Institute of Technology, Bombay, and University of Wisconsin, Madison.

Ryad Benosman received the MSc and PhD degrees in applied mathematics and robotics from the University Pierre and Marie Curie, in 1994 and 1999, respectively. He is currently an associate professor at the University Pierre and Marie Curie, Paris, France, leading the Natural Computation and Neuromorphic Vision Laboratory, Vision Institute, Paris. His work covers neuromorphic visual computation and sensing. He is currently involved in the French retina prosthetics project and in the development of retina implants and cofounder of Pixium Vision a french prosthetics company. He is an expert in complex perception systems, which embraces the conception, design, and use of different vision sensors covering omnidirectional 360 degree wide-field of view cameras, variant scale sensors, and non-central sensors. He is among the pioneers of the domain of omni-directional vision and unusual cameras and still active in this domain. He has been involved in several national and European robotics projects, mainly in the design of artifcial visual loops and sensors. His current research interests include the understanding of the computation operated along the visual systems areas and establishing a link between computational and biological vision. He has authored more than 100 scientific publications and holds several patents in the area of vision, robotics and image sensing. In 2013, he was awarded with the national best French scientific paper by the publication La Recherche for his work on neuromorphic retinas.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.