Hybrid Neural Network, An Efficient Low-Power Digital Hardware Implementation of Event-based Artificial Neural Network

Amirreza Yousefzadeh[1], Garrick Orchard[2],
Evangelos Stromatias[1], Teresa Serrano-Gotarredona[1], and Bernabe Linares-Barranco[1]

[1]*Instituto de Microelectronica de Sevilla (CSIC and Univ. de Sevilla), Sevilla, Spain Bernabe@imse-cnm.csic.es*
[2]*Singapore Institute for Neurotechnology (SINAPSE) at National University of Singapore garrickorchard@nus.edu.sg*

***Abstract*— Interest in event-based vision sensors has proliferated in recent years, with innovative technology becoming more accessible to new researchers and highlighting such sensors' potential to enable low-latency sensing at low computational cost. These sensors can outperform frame-based vision sensors regarding data compression, dynamic range, temporal resolution and power efficiency. However, available mature frame-based processing methods by using Artificial Neural Networks (ANNs) surpass Spiking Neural Networks (SNNs) in terms of accuracy of recognition. In this paper, we introduce a Hybrid Neural Network which is an intermediate solution to exploit advantages of both event-based and frame-based processing. We have implemented this network in FPGA and benchmarked its performance by using different event-based versions of MNIST dataset. HDL codes for this project are available for academic purpose upon request.**

I. Introduction

In Dynamic Vision Sensors (DVS) each pixel spikes as soon as it detects a change in light intensity. Several different event-based temporal contrast DVSs exist [1] [2] [3] [4]. Son et al. [5] recently presented a 640×480 pixel DVS which consumes a total of 27mW at a data rate of 100keps and 50mW at 300Meps, has better temporal resolution than a 2000fps camera and a wide dynamic range of more than 80db.

New techniques for efficient event processing are gradually being introduced. Synaptic Kernel Inverse Method (SKIM) [6], a new learning method for synthesizing SNNs, achieved 92.87% accuracy on the N-MNIST dataset [7]. Kheradpishe et al. [8] developed a multi-layer SNN equipped with Synaptic Time Dependent Plasticity (STDP), achieving 98.4% accuracy on the MNIST dataset [9] by converting all the MNIST frames to events through intensity to delay conversion. J.H.Lee et al. [10] developed a new method to adapt the famous error backpropagation technique for SNNs, achieving 98.66% accuracy on the N-MNIST dataset.

Even though SNNs are improving, training an efficient SNN for hardware implementation is still an open problem. This can be seen in major research initiatives involving training Artificial Neural Networks (ANNs) in frame-based domains and using trained synaptic weights for their SNN counterparts [11] [12]. Although the results of these works seem promising, they also entail serious disadvantages. Firstly, further parameter optimization is required to map from ANN to SNN, because the parameters in SNNs with Leaky Integrate and Fire (LIF) neurons (for example leak rates, threshold and refractory time) are sensitive despite the trained synaptic weights. Secondly, in these works information is coded in the event rate rather than in the exact timing of events [13] [11], so multiple events are needed to transfer information between neurons, thus increasing power consumption and delay.

One major challenge when implementing Leaky Integrate and Fire neurons is to make sure all the neurons have normal activity [12]. Another problem is premature firing before enough information (events from the previous layer) is received. Extra logic also needs to be added for inhibitory connections between neurons, to guarantee competition and eliminate high event rates. Additionally, implementing leakage in digital hardware is not a straightforward task and can be expensive (depending on accuracy).

This paper adopts a hybrid approach combining features of non-spiking synchronous Artificial Neural Network (ANN) and asynchronous Spiking Neural Network (SNN). Our Hybrid Neural Network is a hardware implementation of ANN that uses event-based vision sensors as its input as described in Section II. This Hybrid Neural Network has been implemented in FPGA using Hardware Description Language (HDL). We describe the experimental results of this work in Section III. We have used different event-based versions of MNIST dataset to benchmark our implementation. Additionally, to compare our results with state of the art SNN hardware implementations, we have implemented on FPGA a modified version of the SNN that has been trained with the algorithm presented in [10]. A brief conclusion to this paper is provided in Section IV.

II. Proposed Hybrid Neural Network

The proposed Hybrid Neural Network is an ANN that uses a DVS as its input sensor. As mentioned in Section I, a DVS is a power efficient vision sensor and has considerably less latency than conventional frame-based sensors. When something moves in front of a DVS, multiple pixels almost simultaneously generate events that evoke synchrony-based neural coding [14]. In this kind of coding, information is not spike rate coded or spike rank/order coded [13]. In the DVS even though neurons spike asynchronously information is coded in the simultaneous firing of a group of spikes together. To extract information efficiently, we proposed to

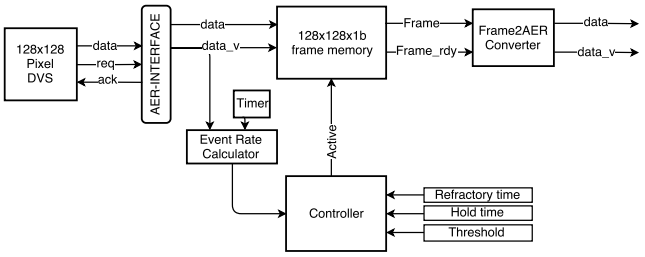

Fig. 1: Block diagram of "frame-maker" hardware implementation. Size of frame-memory should be equal to the size of DVS or smaller (in case of subsampling DVS pixels)

process groups of events that are generated close together in time rather than individual events. There is some evidence that this kind of processing also takes place in the biological cortex [15]. A similar approach has been used in some efficient hardware implementations [16] [17]. A circuit called "frame-maker" was therefore designed to group the events that occurring close together in time into a packet (equivalent to a frame in conventional image processing). By using this "frame-maker" after the DVS, it was possible to create an automatic adaptive frame-rate camera as our system input.

Fig. 1 shows a simplified block diagram of the "frame-maker" circuit. The "Frame-maker" only captures DVS events when the input event rate is higher than a given threshold. That is to say a meaningful movement in front of the DVS or a saccade occurred. The frame memory block gathers all the events into a packet or binary frame after receiving an "Active" signal from the Controller (Active state). As soon as the controller deactivates this signal, the frame-memory stops registering DVS events and issues a Frame_rdy signal, indicating that the frame is ready for further processing (Non-active state). The controller is a Finite State Machine (FSM) based on a simple algorithm (see Algorithm 1). The "AER-INTERFACE" logic block converts the asynchronous communication protocol of the DVS to a synchronous protocol which is more efficient inside an FPGA [18].

To implement our proposed Hybrid Neural Network in hardware, communications between different layers of implemented feedforward ANN is done by using AER events. In this case, each neuron puts its output value and its address into an AER packet and sends it to the next layer after receiving a specific command. This command is coded in AER format as well. We designed a "Frame2AER Converter" logic block to convert the frames into an AER packet.

To explain how our Hybrid Neural Network works, an example block diagram is shown in Fig.2. First, a frame-maker logic block is implemented to convert DVS events into a frame. In this network, we implemented a 28×28 pixel frame-maker by subsampling DVS pixels prior to storing them in the frame-memory[1]. When a frame is ready to propagate, it will be sent to the next layer. An end of frame

[1]In addition to sub-sampling, 2×2 pixels of each border were also cropped to make frames with the exact size of the original MNIST dataset.

Algorithm 1 Frame-maker Controller algorithm. Refractory (Ref) time and Hold time are the parameters that depend on the nature of stimulus and parameters of DVS.

C1 $\leftarrow$ (Event Rate $\geq$ Threshold)
C2 $\leftarrow$ (time $\geq$ Last Non-active time + Ref time)
C3 $\leftarrow$ (Event Rate $<$ Threshold)
C4 $\leftarrow$ (time $\geq$ Last Active time + Hold time)
procedure
Non-active State:
    if *C1 & C2* **then**
        Next State $\leftarrow$ *Active State*
    end if
Active State:
    if *C3 & C4* **then**
        Next State $\leftarrow$ *Non-active State*
    end if
end procedure

(EOF) command event will be generated at the end of each frame.

After the frame-maker, the neural network structure should be implemented. Neurons in the Hybrid Neural Network are using the Rectified Linear unit (ReLu) as activation function [19] which can be efficiently implemented in hardware. When a layer of the neural network receives AER events from the previous layer, it will update the neurons that are connected to that specific input. After receiving an EOF command from the prior layer, each neuron with a positive membrane value sends one event to the next layer and resets to zero. Finally, an EOF event will be generated for the next layer. In this scheme neurons in a layer need to be synchronized with each other, but there is no need for synchronization between different layers. This feature (no need for global synchronization) makes it easier to implement this network in massively parallel platforms like SpiNNaker [20].

To implement the ANN we used 4-bit synaptic weights and neuron membranes. Consequently, each event contains the source address and a 4-bit parameter indicating the membrane voltage of the source neuron. Rather than using extra bits in the AER packet to encode the neurons output, we could have used exact intensity to delay conversion (as in [8]). However, intensity to delay conversion needs to sort all neuron's membrane voltages for each layer and send them out in AER links with exact timing. Sorting thousands of numbers may decrease design performance, so we decided not to use temporal coding in this design. Casting neuron membrane values to 4-bit reduces the number of non-zero neuron outputs that need to be transmitted to the next layer.

In this work, we have implemented configurable cores for fully connected and convolutional processing. Fig.2 illustrates our implemented Hybrid Neural Network for handwriting digit recognition. It contains three convolutional processors with a kernel size of 5×5 in the first layer of the neural network. A 2×2 sub-sampling layer after convolutional cores

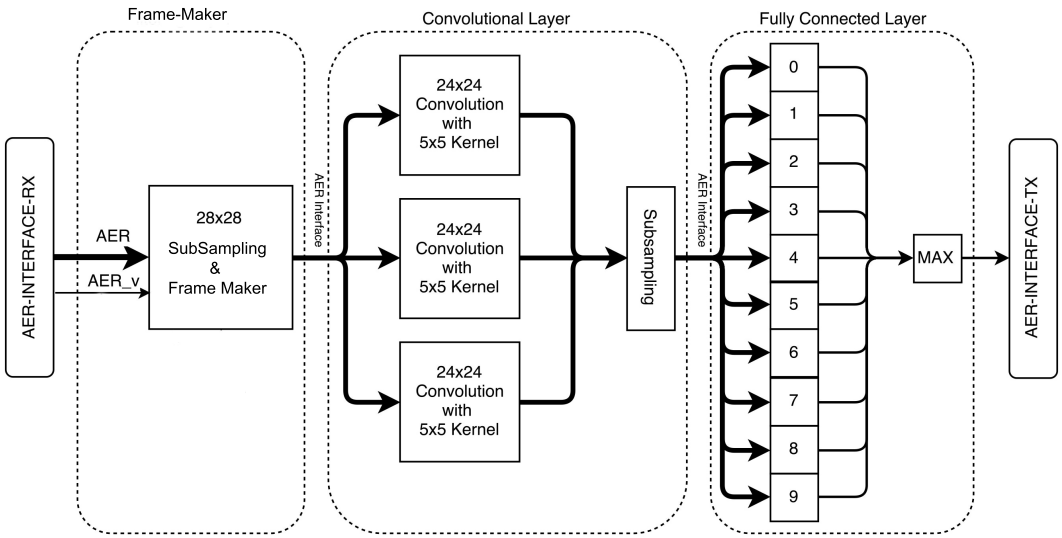

Fig. 2: Block Diagram of FPGA implementation of HybridNet

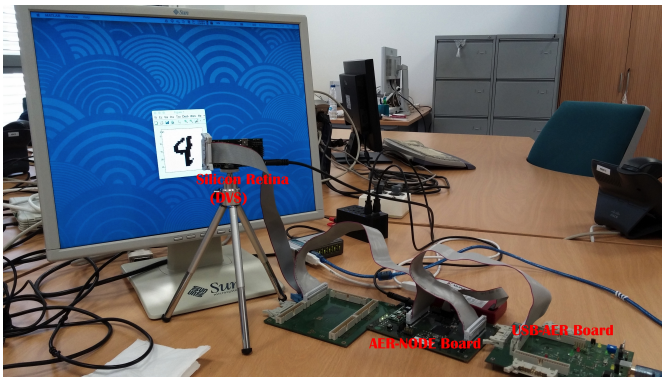

Fig. 3: Hardware setup for real-time processing with DVS. A video of the real-time demonstration is available in [22].

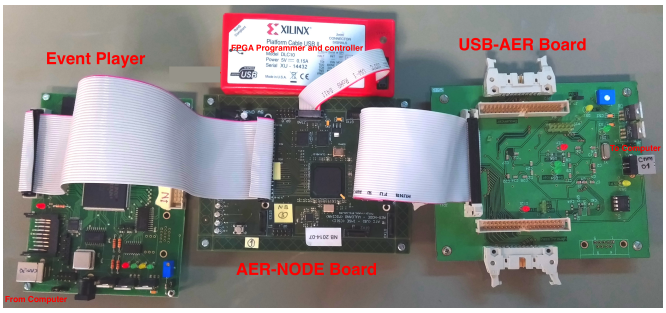

Fig. 4: Hardware setup for processing event-based MNIST dataset by using pre-recorded events in event-player [23].

detects the most activated neuron. The last layer contains ten fully connected neurons. After processing all the input events, a MAX logic operation finds the most activated neuron and presents it as the predicted digit. To train this network, we used the TensorFlow software library [21] and downloaded the synaptic weights to the FPGA after training with 4-bit precision. In TensorFlow, inputs of ANNs are frames. To convert events of neuromorphic datasets to a frame, we have used a software model of the "Frame-maker" logic block.

III. Results

As mentioned earlier, we implemented a small Hybrid Neural Network in FPGA for MNIST dataset recognition. Fig.3 shows our hardware setup with a DVS as the input sensor, an AER-NODE board [24] with Xilinx SPARTAN-6 for implementation of the Hybrid Neural Network and a USB-AER (USBAERmini2) Board[23] for sending events back to a computer in real-time. To test the system with a pre-recorded dataset, rather than using a DVS we used an event player board [23], which played the recorded events in real-time as shown in Fig.4.

To report the accuracy of the network implemented in Fig. 2 we used three different event-based MNIST datasets. First, we used the artificial conversion of MNIST to events by generating one event per non-zero pixels from the MNIST dataset samples and no event for others. We called this dataset synthetic e-MNIST.

The second dataset that we used was the FLASH-MNIST-DVS dataset. The FLASH-MNIST-DVS dataset [25] is a full recording of the MNIST dataset obtained by flashing MNIST digits with a monitor in front of the IMSE-DVS [3]. This dataset is very similar to the artificial conversion of MNIST to events, but it also includes practical noise and statistics from a real DVS.

The last dataset is N-MNIST [7]. This dataset is captured by mounting the ATIS sensor [2] on a motorized pan-tilt unit and having the sensor move while it views the MNIST samples on an LCD monitor. N-MNIST contains three saccades for each MNIST sample. In our Hybrid Neural Network each saccade can be captured as a frame. We have trained one independent Hybrid Neural Network for each set of saccades. Therefore we implemented three Hybrid Neural

TABLE I: The accuracy of FPGA implementation for two-layer Hybrid Neural Network using event-based datasets

Dataset	Train Accuracy	Test accuracy
Synthetic e-MNIST	97.91%	97.09%
FLASH-MNIST-DVS	97.99%	96.80%
N-MNIST	96.59%	96.23%

TABLE II: Comparison of energy efficiency between the proposed approach and the other two famous neromorphic platforms

Platform	Accuracy	Power consumption	Frame-rate
Hybrid-NN on FPGA	97.09%	7uJ per image	58000fps
IBM-TrueNorth [28]	95%	4uJ per image	1000fps
SpiNNaker [29]	95%	6mJ per image	50fps

Networks in parallel (one for each saccade direction). To report the accuracy, we averaged the predictions of all the three networks for each sample.

Table I shows the accuracy of the Hybrid Neural Network implemented in the FPGA. The Spartan-6 (XC6SLX150T-3) implementation of the network illustrated in Fig.2 works up to a clock frequency of 220MHz. The convolutional layer processes each event in '30' clock cycles (135ns). A fully connected layer needs just one clock cycle for each incoming event. In the case of using one spike per non-zero pixels, each frame will be converted to '125' spikes in average. Therefore convolutional processing takes less than 17us for each frame in average. An additional 3us delay will be added by the fully connected layer and the communication system, so that processing latency for each frame is less than 20us. In a pipelined architecture every 17us a new frame can be processed by the FPGA, resulting in more than 58k MNIST frames per second. This FPGA design consumes approximately 363mW for processing 58k frames per second which is equal to less than 7uJ for each frame. Adding more layers will increase latency and power consumption but will not decrease throughput because the layers are implemented in a pipeline.

Most of the logic blocks do not need to process anything when there is no activity in front of the DVS. This implementation consumes '1270' LUTs (1.4%) and '5' Block-RAMs (1.8%) of the FPGA resources[2]

To compare an SNN implementation in FPGA with our proposed Hybrid Neural Network, we have implemented a small SNN with the algorithm presented in [10] by using the MNIST dataset. In this case, we used the Poisson distribution method to convert MNIST frames to spikes because this network does not work correctly with only one spike per non-zero pixels[3]. This SNN uses LIF neurons and contains ten convolutional populations with 5×5 kernel size in the first layer (after input) followed by a ten fully connected neurons output layer.

The original SNN presented in [10] includes complicated synaptic equations. To train the SNN in software, we used the original proposed SNN. However, in the FPGA dynamic synapses have been replaced by static ones and exponential leakage has been implemented with bit-wise shifts [26]. We have used the same hardware setup as shown in Fig.4 for this experiment. Like in our previous design, for each incoming spike, '30' clock cycles were needed for the convolutional process and one clock cycle for updating the fully connected neurons. This implementation occupied '1500' LUTs and '47' Block-RAMs in our Spartan-6 FPGA, and it consumes approximately 388mW when working at full capacity (7M events per second at 220MHz). In this SNN each non-zero pixel is converted to '43' spikes in average (5k spikes for each frame in average) while we only used one spike per non-zero pixels in our proposed Hybrid Neural Network ('125' spikes for each frame in average). Consequently, this design can process less than 1.4k MNIST frames per second which results in consuming in average 300uJ per frame. The accuracy of the implemented SNN in FPGA for MNIST dataset is 97.35%[4]. Even-though FPGA is normally not as power efficient as ASIC design, Table II shows our FPGA approach has good energy efficiency compared to other famous neromorphic platforms in the same digit recognition task.

IV. Conclusion

In this work, we present a hybrid architecture for hardware implementation of ANN that uses DVS as its input. We introduce an FPGA implementation of the proposed Hybrid Neural Network that can be trained off-line by using conventional deep-learning software tools. We demonstrate that a small two-layer Hybrid Neural Network can reach 97% accuracy for MNIST dataset while consuming 7uJ per frame. Finally, we prove that this network can consumes less power than state of the art SNNs in FPGA because each neuron can fire maximum one time for each input frame.

ACKNOWLEDGMENT

This work was supported in part by EU H2020 grants ECOMODE, by Samsung Advanced Institute of Technology grant NPP, and NEURAM3, and by Spanish grant TEC2015-63884-C2-1-P (COGNET) (with support from the European Regional Development Fund).

References

[1] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, T. Delbruck, Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output, Proceedings of the IEEE 102 (10) (2014) 1470–1484. doi:10.1109/JPROC.2014.2346153.

[2] C. Posch, D. Matolin, R. Wohlgenannt, A qvga 143 db dynamic range frame-free pwm image sensor with lossless pixel-level video compression and time-domain cds, IEEE Journal of Solid-State Circuits 46 (1) (2011) 259–275. doi:10.1109/JSSC.2010.2085952.

[2] A video demonstration of real-time handwrite digit recognition in FPGA using Hybrid Neural Network is available here [22].

[3] Around 43 events per non-zero pixel on average is generated which is recommended in the original article [10].

[4] This is the result of using '10' convolutional cores while in the proposed Hybrid-NN we used only '3' convolutional cores. A video demonstration of this SNN while doing handwritten digit recognition in FPGA is available in [27].

[3] T. Serrano-Gotarredona, B. Linares-Barranco, A $128,\times128$ 1.5sensitivity 0.9vision sensor using transimpedance preamplifiers, IEEE Journal of Solid-State Circuits 48 (3) (2013) 827–838. doi:10.1109/JSSC.2012.2230553.

[4] J. A. Lenero-Bardallo, T. Serrano-Gotarredona, B. Linares-Barranco, A 3.6μs latency asynchronous frame-free event-driven dynamic-vision-sensor, IEEE Journal of Solid-State Circuits 46 (6) (2011) 1443–1455. doi:10.1109/JSSC.2011.2118490.

[5] B. Son, Y. Suh, S. Kim, H. Jung, J. S. Kim, C. Shin, K. Park, K. Lee, J. Park, J. Woo, Y. Roh, H. Lee, Y. Wang, I. Ovsiannikov, H. Ryu, 4.1 a 640x480 dynamic vision sensor with a 9um pixel and 300meps address-event representation, in: 2017 IEEE International Solid-State Circuits Conference (ISSCC), 2017, pp. 66–67. doi:10.1109/ISSCC.2017.7870263.

[6] G. K. Cohen, G. Orchard, S. H. Leng, J. Tapson, R. Benosman, A. van Schaik, Skimming digits: Neuromorphic classification of spike-encoded images, Frontiers in Neuroscience-doi:10.3389/fnins.2016.00184.

[7] G. Orchard, A. Jayawan, G. K. Cohen, N. Thakor, Converting static image datasets to spiking neuromorphic datasets using saccades, Frontiers in Neuroscience 9 (2015) 437. doi:10.3389/fnins.2015.00437.

[8] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, T. Masquelier, STDP-based spiking deep neural networks for object recognition, CoRR abs/1611.01421.
URL http://arxiv.org/abs/1611.01421

[9] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[10] J. H. Lee, T. Delbruck, M. Pfeiffer, Training deep spiking neural networks using backpropagation, Frontiers in Neuroscience.

[11] J. A. Prez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, 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 Transactions on Pattern Analysis and Machine Intelligence 35 (11) (2013) 2706–2719. doi:10.1109/TPAMI.2013.71.

[12] P. U. Diehl, D. Neil, J. Binas, M. Cook, S. C. Liu, M. Pfeiffer, Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing, in: 2015 International Joint Conference on Neural Networks (IJCNN), 2015, pp. 1–8.

[13] R. V. Rullen, S. J. Thorpe, Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex, Neural Computation 13 (6) (2001) 1255–1283. doi:10.1162/08997660152002852.

[14] T. Masquelier, G. Portelli, P. Kornprobst, Microsaccades enable efficient synchrony-based coding in the retina: a simulation study, Scientific Reports 6. doi:0.1038/srep24086.

[15] A. Luczak, B. L. McNaughton, K. D. Harris, Packet-based communication in the cortex, Nature Reviews Neuroscience 16 (12).
URL http://dx.doi.org/10.1038/nrn4026

[16] A. G. Andreou, A. A. Dykman, K. D. Fischl, G. Garreau, D. R. Mendat, G. Orchard, A. S. Cassidy, P. Merolla, J. Arthur, R. Alvarez-Icaza, B. L. Jackson, D. S. Modha, Real-time sensory information processing using the truenorth neurosynaptic system, in: 2016 IEEE International Symposium on Circuits and Systems (ISCAS), 2016, pp. 2911–2911.

[17] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, D. Modha, A low power, fully event-based gesture recognition system, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[18] A. Yousefzadeh, M. Jaboski, T. Iakymchuk, A. Linares-Barranco, A. Rosado, L. A. Plana, S. Temple, T. Serrano-Gotarredona, S. B. Furber, B. Linares-Barranco, On multiple aer handshaking channels over high-speed bit-serial bidirectional lvds links with flow-control and clock-correction on commercial fpgas for scalable neuromorphic systems, IEEE Transactions on Biomedical Circuits and Systems 11 (5) (2017) 1133–1147. doi:10.1109/TBCAS.2017.2717341.

[19] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in: J. Frnkranz, T. Joachims (Eds.), Proceedings of the 27th International Conference on Machine Learning (ICML-10), Omnipress, 2010, pp. 807–814.
URL http://www.icml2010.org/papers/432.pdf

[20] M. M. Khan, D. R. Lester, L. A. Plana, A. Rast, X. Jin, E. Painkras, S. B. Furber, Spinnaker: Mapping neural networks onto a massively-parallel chip multiprocessor (2008) 2849–2856doi:10.1109/IJCNN.2008.4634199.

[21] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).

[22] A. Yousefzadeh, Real time demo, spiking MNIST recognition using DVS with proposed hybrid neural network in FPGA, https://youtu.be/FRqH7kRaBW8 (2016).

[23] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gomez-Rodriguez, L. Camunas-Mesa, R. Berner, M. Rivas-Perez, T. Delbruck, S. C. Liu, R. Douglas, P. Hafliger, G. Jimenez-Moreno, A. C. Ballcels, T. Serrano-Gotarredona, A. J. Acosta-Jimenez, 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 Transactions on Neural Networks 20 (9) (2009) 1417–1438. doi:10.1109/TNN.2009.2023653.

[24] T. Iakymchuk, A. Rosado, T. Serrano-Gotarredona, B. Linares-Barranco, A. Jimnez-Fernndez, A. Linares-Barranco, G. Jimnez-Moreno, An aer handshake-less modular infrastructure pcb with x8 2.5gbps lvds serial links, in: 2014 IEEE International Symposium on Circuits and Systems (ISCAS), 2014, pp. 1556–1559.

[25] FLASH-MNIST-DVS dataset, http://www2.imse-cnm.csic.es/caviar/MNISTDVS.html.

[26] A. Yousefzadeh, T. Masquelier, T. Serrano-Gotarredona, B. Linares-Barranco, Hardware implementation of convolutional stdp for on-line visual feature learning, in: 2017 IEEE International Symposium on Circuits and Systems (ISCAS), 2017, pp. 1–4.

[27] A. Yousefzadeh, Real time demo, spiking neural network in FPGA for handwritten digit recognition,, https://youtu.be/U-mrRcaQkpI (2016).

[28] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, D. S. Modha, Backpropagation for energy-efficient neuromorphic computing, in: C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems 28, Curran Associates, Inc., 2015, pp. 1117–1125.

[29] E. Stromatias, D. Neil, F. Galluppi, M. Pfeiffer, S. C. Liu, S. Furber, Scalable energy-efficient, low-latency implementations of trained spiking deep belief networks on spinnaker, in: 2015 International Joint Conference on Neural Networks (IJCNN), 2015, pp. 1–8. doi:10.1109/IJCNN.2015.7280625.