Conversion of Synchronous Artificial Neural Network to Asynchronous Spiking Neural Network using sigma-delta quantization

Amirreza Yousefzadeh*[1], Sahar Hosseini*[2], Priscila Holanda*[1], Sam Leroux[1], Thilo Werner[1], Teresa Serrano-Gotarredona[2], Bernabe Linares Barranco[2], Bart Dhoedt[1], and Pieter Simoens[1]

[1]*Ghent University-imec, IDLab, Belgium*
[2]*Instituto de Microelectronica de Sevilla (CSIC and Univ. de Sevilla), Sevilla, Spain*

***Abstract*— Artificial Neural Networks (ANNs) show great performance in several data analysis tasks including visual and auditory applications. However, direct implementation of these algorithms without considering the sparsity of data requires high processing power, consume vast amounts of energy and suffer from scalability issues. Inspired by biology, one of the methods which can reduce power consumption and allow scalability in the implementation of neural networks is asynchronous processing and communication by means of action potentials, so-called spikes. In this work, we use the well-known sigma-delta quantization method and introduce an easy and straightforward solution to convert an Artificial Neural Network to a Spiking Neural Network which can be implemented asynchronously in a neuromorphic platform. Briefly, we used asynchronous spikes to communicate the quantized output activations of the neurons. Despite the fact that our proposed mechanism is simple and applicable to a wide range of different ANNs, it outperforms the state-of-the-art implementations from the accuracy and energy consumption point of view. All source code for this project is available upon request for the academic purpose[1].**

I. Introduction

The current state of the art hardware implementations of ANNs is not close to the performance of the biological brain concerning the trade-off of power consumption, accuracy, and speed. The human brain contains approximately 86 billions of neurons and 150 trillions of synaptic connections while only consuming around 20W [1]. Visual processing amounts to around 30% and the audio processing consumes around 3% of our brain [2].

Only small fractions of neurons in the brain are active simultaneously and due to asynchronous processing, they mostly consume energy when there is an event to be processed. In the current fully synchronous implementation of ANN algorithms, power consumption is a serious issue and communications between processing units are expensive and not scalable. On the other hand, routing a high-frequency clock signal between distant places in a distributed system is difficult and may be only possible by consuming considerable amount of energy. Therefore, asynchronous communication and processing is the key feature of scalable neuromorphic platforms [3] [4] [5] [6] [7]. The asynchronous activity allows separate processing units to work independently without sharing a clock signal. An asynchronous neuromorphic platform can be scaled up easily without losing performance.

Our focus in this work is on the introduction of a hardware efficient and straightforward method to implement an ANN in an asynchronous way. We aim to keep the method as generic as possible to be compatible with several kinds of ANNs.

In a Spiking Neural Network (SNN) with Integrate&Fire neurons, each neuron has a membrane potential which changes upon the integration of the inputs from other neurons over time. This is somewhat similar to an activation function used in ANNs except the fact that the membrane potential of a spiking neuron evolves over time (dynamic). If a neuron's membrane potential reaches a pre-defined threshold, an output spike is fired and propagated downstream to all its connected neurons. Otherwise, the spiking neuron remains silent and does not provide any output. On the other hand, the output of an ANN neuron is continuous as each ANN unit provides a value in a given range according to its specific activation function. To convert an ANN to SNN, the key question is how to map continuous outputs of ANN units to the activity of spiking neurons.

The straightforward and most used method to code information in spikes is called rate coding [8]. In this method, the output of an ANN unit is mapped to the average firing rate of a spiking neuron [9]. Even-though using frequency of firing as an output of neuron makes the conversion mathematically exact, a few problems with this method makes it less interesting for practical applications.

First of all, because the equations are based on the average firing rate of neurons, several spikes per neuron are needed for the spiking network to become stable. This will increase the needs for processing and communication in a neuromorphic hardware and results in higher energy consumption. The time needed to stabilize rate coding also increases the minimum latency to process an input. These overheads are contradictory to the potential efficiency of SNN because low latency processing and low power consumption are the features that make SNN inference more interesting than their ANN counterparts [10]. The second problem with rate coding comes from the fact that not all architectural elements of modern ANNs can be converted efficiently with rate coding (like biases and max-pooling, etc) [9] [11].

Another famous type of coding information in spikes is temporal coding [8] or Time To First Spike (TTFS) coding [12]. In this coding, information is coded in the exact firing time of spikes. Using this type of coding is not trivial for converting an already trained ANN to SNN. Therefore in previous works, researchers tried to modify the deep learning methods and train the network directly with spiking neurons [13] [14]. For example H. Mostafa [14] and B. Rueckauer et al. [12] presented a learning method where information in spiking neurons are coded in the time of first firing, so each neuron can fire at maximum one time.

This method shows great efficiency from the latency point of view. Additionally, the total number of spikes in this method is very small. However, there are some disadvantages. First, this method is not mature yet and does not perform well in deeper networks. Furthermore, there are some overheads in implementing these networks. For example, in the mentioned works [12] [14], additional to neuron state, synapses are also state-full. This means that the arrival of each spike equals a leaky current injection to the neuron which adds complexity during hardware implementation.

*Amirreza Yousefzadeh, Sahar Hosseini and Priscila Holanda contributed equally in this work
[1]Bernabe@imse-cnm.csic.es

Additionally, the learning mechanism is not straightforward and conversion is not exact, i.e. it is not possible to convert an already trained ANN to SNN using this method. Moreover, same as before, several elements of ANN architecture cannot be converted to the spiking model.

II. Proposed Algorithm

To map a pre-trained ANN to an SNN, we chose a method which is energy efficient, comprehensive and easy to implement. In our scheme, the information is coded in the exact number of spikes. This type of coding is known as spike count coding [8].

In this scheme, there is a one-to-one mapping of ANN neurons to SNN neurons (both architectures have the same number of neurons and synapses). We focus on the fact that an ANN with quantized activation function can be equivalent to an SNN under some assumptions. Fig. 1 shows the ReLU activation function (yellow) which is commonly used in deep learning algorithms to calculate the output of neurons.

The blue and orange lines show the hysteresis quantized version of ReLU activation with a quantization level of one. The quantization level is the step size of the quantizer function. The difference between an input value and its quantized value is referred to as the quantization error and is always smaller than the quantization level.

The reason that we used hysteresis quantization (not direct quantization) is because we found out that direct quantization results in an excessive firing activity of spiking neurons. In direct quantization, when the input (X) is somewhere near the transition point of two quantization levels, small variations/oscillations in X may result in several big changes in the quantized Y which is not desirable. The output of the "Hysteresis Quantizer" depends not only on the current input value but also on the previous value of its output. For example in Fig.1 it is shown that if the current input is '1.1' depending on the previous output of quantizer, current output can be either '2' or '1'. Eq. 1 formulates the hysteresis quantization method where the quantization level is '1'.

$$out_{new} \leftarrow \begin{cases} out_{old} + 1, & (input - out_{old}) > +1 \\ out_{old} - 1, & (input - out_{old}) < -1 \\ out_{old}, & otherwise \end{cases} \quad (1)$$

In this work, we propose to use spikes to communicate the quantized ReLU output value very similar to sigma-delta modulation [15]. Choosing the quantization level is a trade-off as a coarser quantization results in higher quantization error but will generate less spikes[2]. In this work, we used the same threshold for all the neurons.

As mentioned above, unlike a neuronal unit in an ANN, the membrane potential of a spiking neuron dynamically changes over time by receiving asynchronous pre-synaptic spikes. In our current scheme, a pre-synaptic spike charges the neuron immediately. No leakage is implemented, and the membrane potential of all neurons is reset for each new input frame. Additionally, no refractory period is needed which makes the hardware implementation easier.

In this scheme, besides the membrane potential V_{mem}, each neuron has another state which counts how many times the neuron has fired. This value is called quantized membrane potential for a reason that will be explained later. The initial value of V_{mem} after reset is equal to the bias of the corresponding ANN unit and the initial value of V_{mem_q} (quantized V_{mem}) is zero:

$$\begin{aligned} V_{mem}(t=0) &= bias \\ V_{mem\_q}(t=0) &= 0 \end{aligned} \quad (2)$$

Each neuron has a fixed and pre-determined threshold which is equivalent to the quantization level of the ANN activation function. A spiking neuron fires when the difference between its V_{mem} and its V_{mem_q} becomes greater than the threshold. Unlike conventional spiking neurons, our proposed neuron model does not reset after firing[3]. Therefore, after receiving all the presynaptic spikes, the membrane potential of the neurons should converge to the activation level of their ANN counterparts.

Note that the membrane potential does not always increase but because of negative weights may as well decrease. To efficiently handle this fact, we introduced negative spikes. Eq.3 describes how the membrane potential of a neuron updates after receiving a spike from input i.

$$V_{mem} \leftarrow V_{mem} + (W_i \times Sign) \quad (3)$$

where W_i is the weight of input i. Having spikes with a sign bit is inspired by the Dynamic Vision Sensor (DVS) [18] and can be efficiently implemented in neuromorphic platforms [19] without considerable overhead.

Algorithm 1 describes the process of updating V_{mem}, V_{mem_q} and firing spikes when the ANN activation function is ReLU[4].

From Algorithm 1, it is intuitive that V_{mem_q} always follow the quantized value of V_{mem}.

An advanced Artificial Neural Network may use several different methods to increase accuracy and learning performance and new methods are emerging as a result of intensive research. To convert the ANN to asynchronous SNN using the proposed method, only the communication side should be quantized and other processing parts may remain the same.

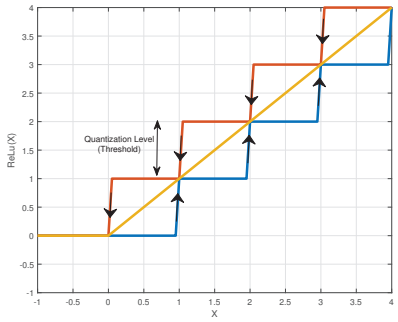


Fig. 1: Hysteresis quantization of ReLu activation function with quantization level (Threshold) of '1'. The yellow line shows ReLu activation function. Going a level up is done with blue lines and going down is done with orange lines.

III. Results

We have applied the proposed method using the MNIST [20] dataset to be able to compare with other state-of-the-art results. Initially, we have trained an ANN for the MNIST dataset using standard stochastic gradient descent. For conversion of the ANN

[2] An extreme case is using an ANN with binary activation [16] [17].

[3] It only resets if a new input (frame) is presented to the network

[4] In this paper, we focus on ReLu activation function ($ReLu(V_{mem}) = max(V_{mem}, 0)$) but the proposed method can apply to other types of quantized activation function as well.

Algorithm 1 Update spiking neuron with Hysteresis Quantization (threshold is normalized to one)

for *each incoming spike* **do**
  $V_{mem} \leftarrow V_{mem} + (W_i \times Sign)$
  $Diff = ReLu(V_{mem}) - V_{mem\_q}$
  if $Diff \geq 1$ **then**
    Fire a positive spike
    $V_{mem\_q} \leftarrow V_{mem\_q} + 1$
  end if
  if $Diff \leq -1$ **then**
    Fire a negative spike
    $V_{mem\_q} \leftarrow V_{mem\_q} - 1$
  end if
end for

network to SNN, the weights and biases are used in the equivalent SNN. To do the conversion we need to define the threshold (equivalent to quantization level) and also a method to convert input frames to spikes.

We decided to convert the input frames to spikes using maximum one spike per pixel. This means each input pixel can fire at most once. We decided to send the pixels with higher intensity earlier and use "linear intensity to delay" coding [21]. Additionally, pixels with intensity equal or less than '0.2' will not fire at all (therefore, the quantization threshold for input frame is '0.2'). Using this method, each MNIST frame is converted to a spike train with duration of '0.8' TU[5]. We have trained a 4-layer convolutional network ($16C5 - 2MP2 - 8C5 - 2MP2 - 256FC[50\%Dropout] - 10FC$)[6]. Table I shows the results of the proposed method for the MNIST dataset.

In Table I, the Quantized ANN (Q-ANN) networks have the same weights and biases as ANN but the activations are quantized (direct quantization without hysteresis) with the same quantization level of SNN. However, the quantization method in SNN is hysteresis which is not exactly similar to quantized ANN. Additionally, the input of quantized ANNs is also quantized with 1 bit.

To measure the accuracy of the SNNs, we counted the number of output spikes (considering the sign of spikes) and the maximum number was assumed as the proposed classification of the network. Also, we reported the total number of firing spikes as a performance metric in Table I. "Avg Num Spikes" reports the average number of spikes per input frame per neuron in the network.

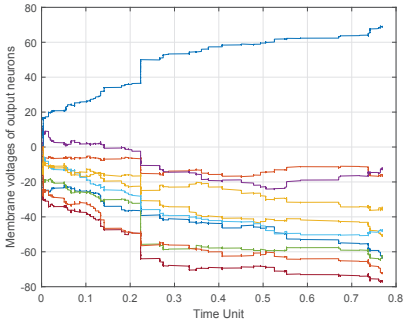
In Table I, we also added the results from other state-of-the-art works on supervised training of Spiking Neural Networks. Most of the previous works only reported the accuracy but not the number of spikes. Therefore, we used the recently introduced and very useful SNN toolbox[7] by B. Rueckauer et al. [9] to reproduce some of the previous works[8]. Our conversion method does not have time-step and simulation finishes after one time presentation of all the input spikes. However, in the case of rate coding where the intensity of the input pixels will be converted to the firing rate, there is not a

[5]Time Unit (TU) is a unit of time and depends on the real-time hardware processing capability. For simulation purposes, a TU can be a second, millisecond, microsecond or so without any practical effect.

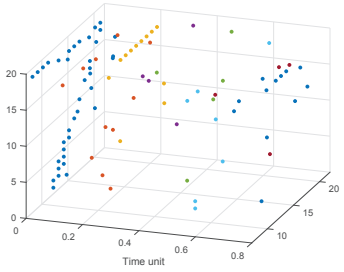
[6]xCy is a convolutional layer with x filters and kernel size of $y \times y$ and xFC is a fully connected layer with x neurons. $xMPy$ is a Max-Pooling layer with kernel size of $x \times x$ and stride of y. For convolutions, we keep the input and output size equal.

[7]http://sensors.ini.uzh.ch/news_page/snn-conversion-2017.html

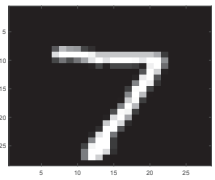
[8]All the codes (including the reproduction of other works) are available upon request



(a)

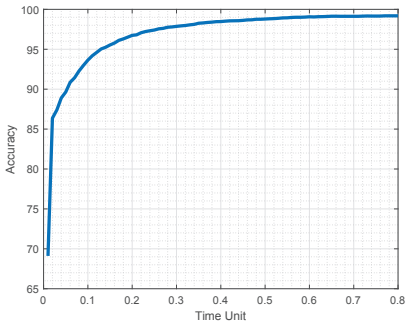


(b)

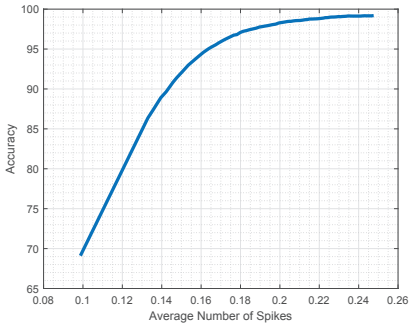


(c)

Fig. 2: (a) Membrane voltage of output neurons over time when presenting the first test image of MNIST. (b) spiking representation of first test image of MNIST over time (c) First test image of MNIST in a frame



(a)



(b)

Fig. 3: (a) Accuracy of Spiking Neural Network versus time for MNIST dataset. (b) Accuracy of Spiking Neural Network versus average number of firing spikes for all the neurons

TABLE I: Accuracy of quantized ANN and equivalent SNN for MNIST dataset.

	Model	Acc ANN	Acc Q-ANN	Acc SNN	Num Neurons	Num Synapses	Avg Num Spikes (per frame)
This work(thr=1.0)	**2xConv-2xFC**	**99.21%**	**98.81%**	**99.19%**	**15k**	**100k**	**0.25**
This work(thr=2.0)	2xConv-2xFC	99.21%	96.44%	97.26%	15k	100k	0.08
[9](Reproduced, 25 time steps)	2xConv-2xFC	99.21%	—	99.18%	15k	100k	0.48
[9](Reproduced, 15 time steps)	2xConv-2xFC	99.21%	—	98.80%	15k	100k	0.26
[12](Temporal coding)	3xConv-2xFC	98.96%	—	98.57%	7.7k	1.2M	0.13
[12](Temporal coding)	2xFC	98.50%	—	98.35%	1.4k	476k	0.07
[22](Temporal coding)	2xFC	98.50%	—	96.98%	1.4k	476k	0.10
[23](Rate coding)	2xConv-1xFC	99.14%	—	99.1%	5.1k	50k	19.25
[9](Rate coding)	—	99.44%	—	99.44%	8k	1.2M	—
[24](Rate coding)	1xConv-1xFC	98.3%	—	98.32%	9.5k	88k	—
[25](Rate coding)	2xConv-2xFC	—	—	99.42%	20.7k	0.6M	—
[14](Temporal coding)	2xFC	—	—	97.55%	1.5k	635k	—

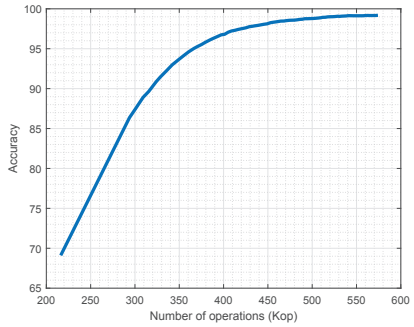


Fig. 4: Accuracy versus the average number of operations for MNIST dataset.

definite time for the end of the input presentation. In this case, by the longer presentation of the input, the accuracy increases until the network becomes completely stable. On the other hand, longer input presentation results in more number of spikes in average, resulting in a trade-off. To demonstrate this trade-off, we reported the reproduced results with different time-steps.

Fig. 2 (a) illustrates the membrane voltage of the output neurons over time for one of the MNIST frames. From Fig. 2(a) it can be seen that the classification results are correct much before the presentation of all the input spikes. Therefore, we measured the accuracy of the SNN over time and over the number of fired spikes which is plotted in Fig.3.

In an ANN, the number of operations per each frame is fixed[9] while in an SNN it depends on the content of the frame and the desired accuracy. For the current network, in ANN each MNIST frame needs 455K MAC (Multiply-Accumulation) operations[10]. If we consider a MAC operation equal to two accumulations[11], ANN needs 911K operations per each frame. In the case of SNN, Fig.4 shows the average number of accumulation operations (synaptic update) which is needed to achieve a specific accuracy[12]. We can see that the SNN needs less than 600K operations to achieve the same accuracy as the corresponding ANN. It should be noted that for a more exact calculation, the other operations like memory read/write and spike communications should be considered.

[9]Though there are some hardware accelerators of ANN which are optimized for sparse activity [26], their implementation needs extra logics for sparsity controller.

[10]$(28 \times 28 \times 16 \times 5 \times 5) + (14 \times 14 \times 8 \times 5 \times 5) + (256 \times 7 \times 7 \times 8) + (10 \times 256)$

[11]Normally a multiplication is more expensive than several additions in hardware.

[12]The proposed Spiking Neural Network does not need multiplication.

IV. Conclusion

In this work, we presented a new method to convert a synchronous ANN to an asynchronous SNN with minimum overhead. Even though our goal was not to mimic biological neural networks completely, we used the sparsity and asynchronous communication features of bio-inspired implementations. We aimed to offer a better implementation of synchronous ANN since a direct ANN implementation is not easily scalable and won't easily exploit the sparsity of the network for efficient power consumption.

Even though the works that use TTFS coding [12] [14] [22] show great efficiency from latency and power consumption point of view, our proposed method has the advantage to be more accurate and scalable for larger networks. Additionally, our proposed method offers a straight-forward conversion of any ANN modules thanks to its simplicity. In comparison with rate-coding conversions, the proposed method results in less number of spike activity (therefore less power consumption and latency) while keeping the same accuracy.

ACKNOWLEDGMENT

This work was supported in part by IMEC-IDLab, and by EU H2020 grant 687299 NEURAM3, and by Spanish grant from the Ministry of Economy and Competitivity TEC2015-63884-C2-1-P (COGNET) (with support from the European Regional Development Fund). The authors would like to thank iMind institute for providing the tools and infrastructure for deep learning.

References

[1] J. W. Mink, R. J. Blumenschine, D. B. Adams, Ratio of central nervous system to body metabolism in vertebrates: its constancy and functional basis, American Journal of Physiology-Regulatory, Integrative and Comparative Physiology 241 (3) (1981) R203–R212.

[2] D. Grady, The vision thing: Mainly in the brain, Discover.

[3] G. Indiveri, B. Linares-Barranco, T. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Hfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. SAGHI, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, K. Boahen, Neuromorphic silicon neuron circuits, Frontiers in Neuroscience 5 (2011) 73. doi:10.3389/fnins.2011.00073.

[4] M. Davies, N. Srinivasa, T. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. Weng, A. Wild, Y. Yang, H. Wang, Loihi: A neuromorphic manycore processor with on-chip learning, IEEE Micro 38 (1) (2018) 82–99. doi:10.1109/MM.2018.112130359.

[5] S. B. Furber, F. Galluppi, S. Temple, L. A. Plana, The spinnaker project, Proceedings of the IEEE 102 (5) (2014) 652–665.

[6] 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, D. S. Modha, A million spiking-neuron integrated circuit with a scalable communication network and interface, Science 345 (6197) (2014) 668–673. doi:10.1126/science.1254642.

[7] A. Yousefzadeh, G. Orchard, E. Stromatias, T. Serrano-Gotarredona, B. Linares-Barranco, Hybrid neural network, an efficient low-power digital hardware implementation of event-based artificial neural network, in: 2018 IEEE International Symposium on Circuits and Systems (ISCAS), 2018, pp. 1–5. doi:10.1109/ISCAS.2018.8351562.

[8] 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.

[9] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, S.-C. Liu, Conversion of continuous-valued deep networks to efficient event-driven networks for image classification, Frontiers in Neuroscience 11 (2017) 682.

[10] C. Farabet, R. Paz, J. Perez-Carrasco, C. Zamarreo, A. Linares-Barranco, Y. LeCun, E. Culurciello, T. Serrano-Gotarredona, B. Linares-Barranco, Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel convnets for visual processing, Frontiers in Neuroscience 6 (2012) 32.

[11] A. Sengupta, Y. Ye, R. Wang, C. Liu, K. Roy, Going deeper in spiking neural networks: VGG and residual architectures, CoRR abs/1802.02627. arXiv:1802.02627.

[12] B. Rueckauer, S. C. Liu, Conversion of analog to spiking neural networks using sparse temporal coding, in: 2018 IEEE International Symposium on Circuits and Systems (ISCAS), 2018, pp. 1–5.

[13] J. Lee, T. Delbrück, M. Pfeiffer, Training deep spiking neural networks using backpropagation, CoRR abs/1608.08782.

[14] H. Mostafa, Supervised learning based on temporal coding in spiking neural networks, IEEE Transactions on Neural Networks and Learning Systems (2018) 1–9.

[15] B. E. Boser, B. A. Wooley, The design of sigma-delta modulation analog-to-digital converters, IEEE Journal of Solid-State Circuits 23 (6) (1988) 1298–1308. doi:10.1109/4.90025.

[16] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, Xnor-net: Imagenet classification using binary convolutional neural networks, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), Computer Vision – ECCV 2016, Springer International Publishing, Cham, 2016, pp. 525–542.

[17] S. Leroux, S. Bohez, T. Verbelen, B. Vankeirsbilck, P. Simoens, B. Dhoedt, Transfer learning with binary neural networks, CoRR abs/1711.10761. arXiv:1711.10761.

[18] T. Serrano-Gotarredona, B. Linares-Barranco, A 128×128 1.5% contrast sensitivity 0.9% fpn 3us latency 4mw asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers, IEEE Journal of Solid-State Circuits 48 (3) (2013) 827–838.

[19] A. Yousefzadeh, M. Soto, T. Serrano-Gotarredona, F. Galluppi, L. Plana, S. Furber, B. Linares-Barranco, Performance comparison of time-step-driven versus event-driven neural state update approaches in spinnaker, in: 2018 IEEE International Symposium on Circuits and Systems (ISCAS), 2018, pp. 1–4. doi:10.1109/ISCAS.2018.8350990.

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

[21] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, T. Masquelier, STDP-based spiking deep neural networks for object recognition, CoRR abs/1611.01421.

[22] H. Mostafa, B. U. Pedroni, S. Sheik, G. Cauwenberghs, Fast classification using sparsely active spiking networks, in: 2017 IEEE International Symposium on Circuits and Systems (ISCAS), 2017, pp. 1–4. doi:10.1109/ISCAS.2017.8050527.

[23] 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.

[24] E. Stromatias, M. Soto, T. Serrano-Gotarredona, B. Linares-Barranco, An event-driven classifier for spiking neural networks fed with synthetic or dynamic vision sensor data, Frontiers in Neuroscience 11 (2017) 350. doi:10.3389/fnins.2017.00350.

[25] Y. Wu, L. Deng, G. Li, J. Zhu, L. Shi, Spatio-temporal backpropagation for training high-performance spiking neural networks, Frontiers in Neuroscience 12 (2018) 331.

[26] B. Moons, M. Verhelst, An energy-efficient precision-scalable convnet processor in 40-nm cmos, IEEE Journal of Solid-State Circuits 52 (2017) 903–914.