EVENT-BASED REAL-TIME HAND GESTURE RECOGNITION USING SPIKING NEURAL NETWORK

Van Khoa LE and Sylvain Bougnoux

IMRA Europe S.A.S., 220 Rue Albert Caquot, 06904 Sophia-Antipolis

ABSTRACT

Deep learning represents the state of the art in many machine learning and computer vision problem. The core of this technology is the analog neural network (ANN) composed of multiple convolution and pooling layers. Unfortunately, such system demands massive computational power thus consuming a lot of energy and therefore causing negative effect to the environment. On one hand, human brain is known to be much more energy efficient, so the spiking neural network (SNN) was created to replicate the brain activity in order to improve the energy efficiency of current deep learning model. On the other hand, the event-based domain based on neuromorphic sensor like event camera made huge progress since last few years and become more and more popular. The data signal flow in spiking neural network is a perfect fit for the output of event camera. Therefore, in this article we built a system based on the combination of event camera and SNN for the real-time hand gesture recognition. We also give an analysis to prove the energy efficiency of this technology compared to the ANN counterpart.

KEYWORDS

Event camera, Spiking Neural Network, Neuromorphic engineering

1. INTRODUCTION

Deep learning has emerged in most of applications and outperforms other machine learning algorithms in cognitive tasks [1]. However, it is computational greedy, which mostly comes from graphic cards consuming a lot of energy. It’s an obstacle for long-term active systems, or edge devices. To reduce the computation costs, many researchers have focused on optimizing the implementation of the neural network. A promising direction is inspired by the fact that human brain is much more computation efficient than GPU, and modern Human Brain Interface devices shown that the communication between synapses in human brain, is based on spike signals. So the event-driven spiking neural network, third generation of deep learning was developed to deal with this kind of information. There are two major differences of SNN, compared to conventional neural network. The first one is that in SNN, inputs are presented as a stream of events, and this stream are computed asynchronously. Each neuron communicates with each other by spike. The second difference is that each neuron in SNN has an internal state to decide whether it will emit a spike to its listening neurons (basically in the next layer).

Moreover, in the event-based computer vision domain, neuromorphic sensors such as event cameras are developed by multiple companies to copy the human retina ability. This kind of sensors generates spikes (events) when it observes a change in the intensity of a pixel. Unlike frame in conventional cameras, this mechanism gives event cameras many advantages such as a high temporal resolution, which could achieve 100 KHz, high dynamic range, and most of all no redundancy in the data. Thus, event camera becomes a perfect fit for SNNs to create an energy efficient computer vision solution. However, even though the development of SNNs and neuromorphic sensors made big progresses since the last few years, they still lack researches that apply these technologies for real world applications. One obstacle is that the access to the hardware is not obvious; another reason is that the SNN are difficult to train due to the non-differentiable characteristic of the spiking neuron's activation function, which is not trainable with classical gradient descents as with ANN.

In this paper, we create a 4-classes hand gesture recognition system based on an event camera and a spiking neural network. Our hand gesture dataset recorded by the event camera is used to train and evaluate several neural network architectures. In particular, a comparison of many aspects between event-based SNN and ANN is made, where energy efficiency is emphasized. We also analyze the effect of different parameters of SNN to the experimental results. Our results show that SNN on event-based hand gesture recognition could be more energy efficient than ANN counterpart in a large factor, especially in long-term active mode.

2. RELATED WORK

Hand gesture recognition is a popular topic in computer vision. In the last few years, the features trained by ANN composed of convolution and pooling layers outperform handcraft features on such task [2], [3]. However, such systems disadvantaged by the data redundancy, lead to many wasteful computation efforts in long-term active system. Plus, the normal frame rate of RGB camera being about 30Hz, information between frames in a dynamic scene is lost.

Due to the growing development of event camera like the DAVIS [4], ATIS [5], many researchers work in the event based vision for multiple tasks such as image segmentation [6], optical flow [7] and object detection [8]. The similar points of these papers are they constructed images from event streams and train ANN models from these images. This approach often works, but it demands a lot of computation cost and the calculation time might not be sufficient for real time applications.

Many training algorithms for SNN were developed in the last few years. In general, there are three main approaches, the first one is modifying the back-propagation by adding differentiable low pass filter [9] or using surrogate gradient [10]. Although the accuracy using this approaches have been improved, but it requires a lot of time for training. The second approach applies the spike-timing-dependent-plasticity (STDP) algorithm [11], a biology inspired, layer-wised, and unsupervised learning method. However, it currently only performs well on small datasets, and still shows limitation in training deep architecture networks. The last approach is to train an ANN counterpart with back-propagation, and then convert the weights for SNN [12]. This approach can be used to train deep networks and can take advantage of mature learning algorithms of ANN.

One of the first work to use event camera for the gesture recognition was presented in [13]. But they implemented their SNN on neuromorphic hardware [14], hardware with non-von Neumann architecture, which is still limited by the time of this article. [15] achieved a good accuracy on a live demonstration with event camera based on ANN, but the energy gain was not studied. For the comparison between SNN and ANN, [16] proposed one, but their network is small, being composed of a single fully connected layer. Therefore its results might not extrapolate for bigger networks composed of multiple convolution and pooling layers.

These previous works gave us the motivation to create a hand gesture recognition system based on an event camera, and a SNN run on a conventional hardware. In general, streams of event from an event camera are captured and converted to count-images. Basically a count-image counts the number of spikes arrived per pixel during a given period. The training and test are done offline. From a dataset of count-images, the ANN model is trained with conventional gradient descent algorithm, while the SNN model is trained using the conversion approach. For inference, the count-images are created and propagated in the trained SNN model in real time. Up to our knowledge, this is the first work to create a real world application combining an event camera and a SNN which can run on real time using GPU, and also the first to give energy focused analysis.

3. HAND GESTURE RECOGNITION WITH ANN

3.1. The dataset

In event-based domain, an event camera creates an event at a pixel when the intensity of that pixel changes sufficiently (instead of capturing the whole frame as in a frame-based camera).An event camera produces a stream of events as output. Each event e_i has four components: (x_i, y_i) is the pixel location, a polarity $p_i \in \{-1,1\}$ indicates the decrement or increment of intensity, and a timestamp t_i:

$$e_i = \{x_i, y_i, p_i, t_i\} \tag{1}$$

The 3D interface in Figure 1 (left) helps visualizing the events stream in spacial-temporal axes.The two axes at the surface are the horizontal and vertical axis of the camera, while the third axis corresponds to timestamp of each event; lastly the color of pixels is the event's polarity.

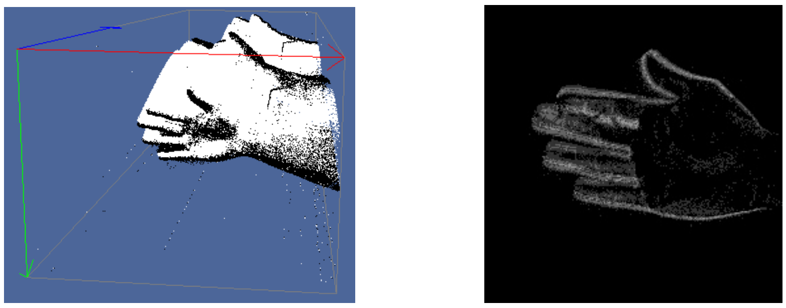

Figure 1: (left) 3D plot of events with the third axis being the time axis. Positive events are plot in white and negative ones are plot in black. (right) Count-image of accumulated events within $\mathbf{10}\ \boldsymbol{ms}$.

Our dataset was recorded with an event camera with a VGA resolution, high dynamic range, and high time resolution. The camera was fixed so there is no background. We first crop a rectangle of 227x227 in the camera's field of view, and then we construct the count-image with the events of this area.

In general, there are two approaches to build the count-image, by fixing: the total number of events; or the sample rate. We found that selecting the fixed number of event as in [15] and [17] has the disadvantage that in real scenario, the event flow is largely unstable, depending for instance on the light or on the scene. Therefore it is difficult to fix this number of event, and we prefer constructing the image by defining a fixed sample rate.

Frames are constructed by accumulating events in the period between each sample. The intensity of each pixel in this image, is the number of events arriving within that period. In real time demonstration, we can always change the intensity increment for each event to adapt the situation. An example of a count-image is presented in Figure 1 (right).

3.2. Training ANN

We trained an ANN models with a dataset of count-images. Five models are presented in Table 1 to verify the effect of different architectures. The images are normalized to the range [0,1] before sending to the network. The softmax loss was applied at the last layer.

Table 1. Five architectures for recognition task.

Text	**Architecture**
Net1	Conv(5x5x6) + Pool + FC(128)
Net2	Conv(5x5x6) + Pool + FC(256)
Net3	Conv(5x5x6) + Pool + Conv(5x5x16) + FC(128)
Net4	Conv(5x5x6) + Pool + Conv(5x5x16) + Pool + FC(128)
Net5	Conv(5x5x6) + Pool + Conv(5x5x16) + Pool + FC(128) + FC(128)

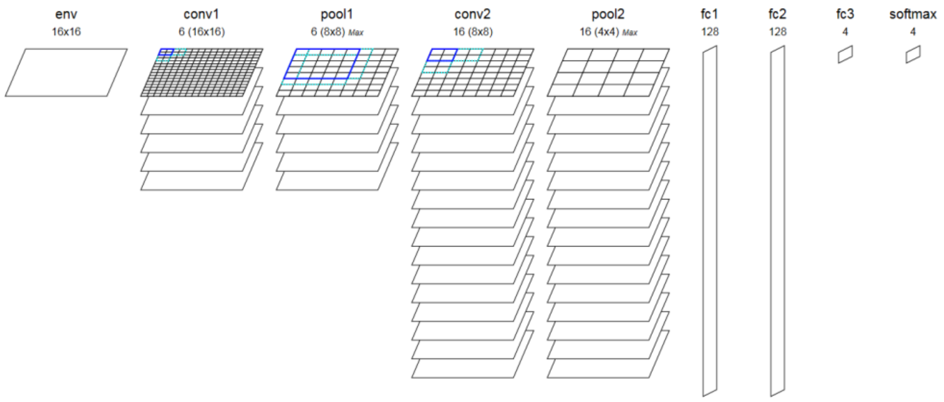

Figure 2: Architecture of Net5.

Figure 2 present the feature map after each layer of the network Net5. The training was implemented using the N2D2 library [18], and trained for 6 million iterations with a batch size 128 and Adam optimization algorithm. Learning rate starts at 0.05, with $decay = 5 * 10^{-4}$, and $momentum = 0.9$. The activation function is ReLU, with a weight initialization with HeFiller approach [19].

4. HAND GESTURE RECOGNITION WITH SNN

4.1. Spike encoding

For the SNN experiment, we generate synthetic spikes to recreate the stream of spikes from the count-image. Several spike coding approaches was presented such as the rate coding, time coding, rank coding, etc [20]. In this work, we use the rate coding [21] to encode the value of the input pixel as the frequency of the spikes.

This encoding leads to an easy conversion from ANN to SNN. Spike trains are created by randomly selecting the intervals between spikes dt from an exponential distribution with the parameter λ as the period. Each event's timestamp is selected by adding to previous event's timestamp the randomly sampled interval dt.The normalized input intensity $x \in [0,1]$ decides the $\lambda \in \{\lambda_{min}, \lambda_{max}\}$ using a hyperbolic interpollation, as in equation 2.

$$\lambda(x) = \frac{1}{\lambda_{max}^{-1} + (\lambda_{min}^{-1} - \lambda_{max}^{-1}) * (1-x)} \tag{2}$$

To simplify the situation, we choose a fixed λ_{max}, and vary λ_{min} for our experiments.

4.2. Integrate and Fire neuron model

A SNN model consists of multiple layers $l \in [0, L]$ each neuron in the network is an Integrate and Fire (IF) neuron with the membrane potential $U_i^l$and threshold θ. The spike simulation time is performed for $t \in [0, T]$.

$$U_i^l(t + \Delta t) = U_i^l(t) - \theta s_i^l(t) + \sum_j w_{ij}^l s_j^{l-1}(t) \tag{3}$$

where w_{ij}^l is the weight between neuron j of layer $l - 1$ and neuron i of layer l. The activation function $s_i^l \in \{0,1\}$ is a trigger function depending on the membrane potential U_i^l. A spike is generate if $U_i^l > \theta$, and $U_i^l$is reduced by an amount of θ. The threshold $\theta$is set to 1.

4.3. Conversion learning

The training of SNN models was done using the conversion method. The main idea of the conversion is that the float values of the activation function ReLU's output of the original ANN can be encoded as firing rates in SNN with the IF neuron [22] and this mapping is improved easily by scaling the weight matrices [12]. In the last layer, each neuron is associated to a class, and the class receiving the more spikes is the predicted class.

4.4. ANN vs SNN in hand gesture recognition

The comparison between ANN and SNN is not an easy task because the two paradigms are different. On one hand, the result of ANN for one specific architecture is constant in term of accuracy or number of computation (flops) for one inference. On the other hand, SNN provide a flexibility to have a trade-off between accuracy-flops, accuracy-latency by tuning some parameters. Such parameters are for instance those of the spike generator, e.g. the firing threshold θin equation 3. The SNN are also influenced, by the nature of the input scene (dynamic or static), and other parameters of the camera. Thus, we keep same architecture and same dataset to make fair comparison.

In hand gesture recognition, the accuracy and inference speed are usually focused. In this work we emphasize the energy and the computation cost comparison between SNN and ANN in order to witness the advantages of SNN for long-term active systems, where energy is limited such as mobile objects, vehicles, etc. Therefore, our objective is to find a balance between accuracy, latency, and energy.

5. EXPERIMENTS

5.1. Setup

Inspired by the live demo in [15], we made a hand gesture recognition system to play Roshambo with SNN on GPU, and the same experiment in ANN for comparison. There are four classes in our dataset: rock, paper, scissor and background. The dataset that we used was captured by the Prophesee event camera generation 3. We constructed the count-image by accumulating events during 10 ms. In this dataset, there are 60.000 images in the training set, and 12.000 images in the test set. The experiments are constructed with the library N2D2 [18], a deep learning library that support spike conversion and multiple evaluation metric. The experiment is done in a laptop core I7, 16Gb RAM and graphic board Geforce 1070 GTX.

5.2. Results

5.2.1. Comparison between ANN and SNN

The first part of the analysis starts with the comparison between ANN and SNN on our test set. For each architecture, we compare the performance between ANN and SNN, and the effect of the spike generation parameter $\lambda_{min}$with three different values of $\lambda_{min} = \{0.05, 0.1, 0.2\}$microseconds (us), where $\lambda_{max} = 1ms$ is fixed.

As the accuracy computation is trivial, we only discuss how the calculation of the number of operation and the energy gain of SNN and ANN were done. The number of operation of ANN is constant for every stimuli, which is also the number of connection in the network. The counterpart of SNN is the average number of spikes flow through the network for all stimuli. We compared the ratio between the number of operation of SNN and ANN and called it operation rate (Opr).

The energy efficiency advantage of SNN compared to ANN is twofold: the energy needed for each operation, and in long-term active mode. An operation in ANN is a multiplication accumulation (MAC), while the counterpart in SNN is only accumulation (ACC) due to the binary spikes. According to [23], the energy required for a MAC is $1.27\ pJ$, and an energy for a ACC operation is $0.03\ pJ$. To estimate the energy gain of SNN to ANN for an inference, we multiply the average number of operations for its corresponding energy factor, and then divide the energy cost of ANN to SNN to have the gain.

Table 2. Comparison between ANN and SNN with different value λ_{min}.

Model	λ_{min}	**Acc (%)**	**Opr**	**Energy gain (×)**
Net1 (ANN)		89.46	1	1
Net1 (SNN)	0.05	84.81	2.96	14.3
	0.1	77.37	1.54	27.5
	0.2	71.44	0.78	54.3
Net2 (ANN)		90.77	1	1
Net2 (SNN)	0.05	87.62	3.25	13
	0.1	82.74	1.69	25.05
	0.2	77.1	0.84	50.4
Net3 (ANN)		92.43	1	1
Net3 (SNN)	0.05	90.4	3.01	14.06
	0.1	88.06	1.91	22.16
	0.2	85.02	0.99	42.76
Net4 (ANN)		93.54	1	1
Net4 (SNN)	0.05	91.15	2.92	14.5
	0.1	88.39	1.82	23.26
	0.2	82.62	0.93	45.52
Net5 (ANN)		93.6	1	1
Net5 (SNN)	0.05	91.92	2.26	18.73
	0.1	89.37	1.71	24.75
	0.2	84.61	1.06	39.93

In the long-term active mode, the system is always active and the background is static. Most of the time there is no activity in front of the camera, ANN system have to do the computation constantly with a fixed frame rate and thus cause many wasteful computations. It is clear that SNN is more energy efficient because most of its neurons stay idle and only few computations are made.

The result is shown in Table 2. In general, ANN models achieve better accuracy than SNN counterparts, and this gap is reduced for bigger networks and higher spike rate. For a specific network, the gap between SNN and ANN changes with different λ_{min}. For example, with $\lambda_{min} = 0.05$, the gap between accuracy of SNN and ANN is smaller than with $\lambda_{min} = 0.2$. This could be explained because with smaller λ_{min}, more spikes are generated, and the SNN network encodes more precisely the float value as in ANN. This explanation is also held for the operation rate, but reverse. We can generally observe that the operation rate is higher than 1, but we have some gain with $\lambda_{min} = 0.2$. However, in the experiments that were made, the energy gain column in Table 2 still shows an impressive gain ($13 - 50\ \times$) of SNN compared to ANN.

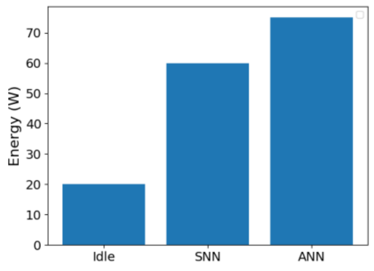

Figure 3:Energy comparison between SNN and ANN when running on laptop.

As this result is theoretical estimation and can only be achieved with a solid implementation, we also measure the energy consumption when running ANN and SNN in live mode on our laptop.The result is in Figure 3, where we first measure the energy of the laptop in idle state, then run the ANN and SNN on the same event stream for 5 min, and the energy is logged every 10 s.The result gives us an energy consumption gain of 20% when comparing SNN to ANN.

5.2.2. Trade-off in SNN system

SNN is different from ANN because it proposes several trade-off. For instance: between accuracy and computation time and between accuracy and number of operations (Figure 4). The experiments are made for 5 architectures with 3 different values of λ_{min}.

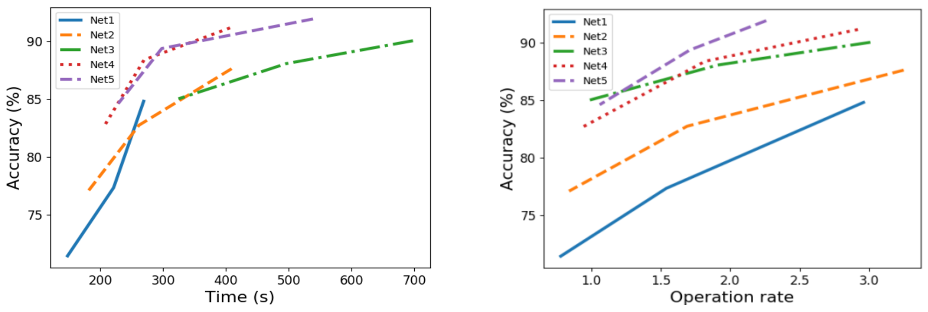

Figure 4: Trade-off between accuracy-computation time (left) and accuracy-operation rate (right).

5.2.3. Real time recognition

The hand gesture recognition system with SNN can run in real time at high frequency. For example, with the Net5 architecture, in the ANN, the computation time is fixed about 30 Hz. In the SNN the variance between a background frame and a gesture frame is huge, it is $500\ Hz$ for background frame and around $30\ Hz$ for gesture frame. This result is explainable because with background, SNN generates very few spikes, thus not sufficient to propagate to deeper layers.

6. CONCLUSIONS

In this work, we presented an energy comparison between ANN and SNN on an event-based real-time hand gesture recognition system. The accuracy of SNN and ANN are comparable, and the flexibility of SNN was witnessed. The energy efficiency of SNN is much better than ANN (13-50×) due to the different operation types between SNN and ANN. Our experiments show that the proposed framework could run a SNN on standard hardware such as CPU/GPU, and canreduce considerably its energy consumption. The SNN system is faster, does less computation in long active mode, and saves a lot of energy in non-dynamic scene. In future works, we will continue to analyze the effect of different layers, the size of the input count-image, and the spike generation method that contributes to better accuracy and better energy consumption

ACKNOWLEDGEMENTS

We would like to thank Prophesee to have kindly supported, and lent us one of their latest camera; and Olivier Bichler from CEA-LIST for his support on the N2D2 library

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015

[2] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, "Hand gesture recognition with 3d convolutional neural networks," in *proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2015, pp. 1–7

[3] O. Koller, H. Ney, and R. Bowden, "Deep hand: How to train a cnn on 1 million hand images when your data is continuous and weakly labelled," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3793–3802.

[4] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240× 180 130 db 3 μs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[5] 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 Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, 2010.

[6] I. Alonso and A. C. Murillo, "Ev-segnet: Semantic segmentation for event-based cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[7] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "Ev-flownet: Selfsupervised optical flow estimation for event-based cameras," *arXiv preprint arXiv:1802.06898*, 2018.

[8] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "Attention mechanisms for object recognition with event-based cameras," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 1127–1136.

[9] C. Lee, S. S. Sarwar, and K. Roy, "Enabling spike-based backpropagation in state-of-the-art deep neural network architectures," *arXiv preprint arXiv:1903.06379*, 2019.

[10] E. O. Neftci, H. Mostafa, and F. Zenke, “Surrogate gradient learning in spiking neural networks,” *arXiv preprint arXiv:1901.09948*, 2019.

[11] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, and T. Masquelier, “Spyketorch: Efficient simulation of convolutional spiking neural networks with at most one spike per neuron,” *arXiv preprint arXiv:1903.02440*, 2019.

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

[13] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza *et al.*, “A low power, fully event-based gesture recognition system,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7243–7252.

[14] 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 *et al.*, “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[15] I.-A. Lungu, F. Corradi, and T. Delbruck, “Live demonstration: Convolutional neural network driven by dynamic vision sensor playing roshambo,” in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–1.

[16] L. Khacef, N. Abderrahmane, and B. Miramond, “Confronting machinelearning with neuroscience for neuromorphic Architectures design,” in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[17] B. Ruckauer, N. K ̈ anzig, S.-C. Liu, T. Delbruck, and Y. Sandamirskaya, ̈ “Closing the accuracy gap in an event-based visual recognition task,” *arXiv preprint arXiv:1906.08859*, 2019.

[18] O. Bichler, D. Briand, V. Gacoin, and B. Bertelone, “N2d2-neural network design and deployment,” *Manual available on Github*, 2017.

[19] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[20] P. Filip and K. Andrzej, “Introduction to spiking neural networks: information processing,” *Learn. Appl*, vol. 71, pp. 409–433, 2011.

[21] D. Heeger, “Poisson model of spike generation,” *Handout, University of Standford*, vol. 5, pp. 1–13, 2000.

[22] A. Tavanaei and A. Maida, “Bp-stdp: Approximating backpropagation using spike timing dependent plasticity,” *Neurocomputing*, vol. 330, pp. 39–47, 2019.

[23] M. Horowitz, “1.1 computing’s energy problem (and what we can do about it),” in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 2014, pp. 10–14