Event-based Regression with Spiking Networks

Elisa Guerrero[1 [0000−0002−8320−0811]], Fernando M. Quintana[1 [0000−0001−5042−9399]], and Maria P. Guerrero-Lebrero[1 [0000−0002−4905−8241]]

University of Cadiz. Spain
elisa.guerrero@uca.es
fernando.quintana@uca.es
maria.guerrero@uca.es

Abstract. Spiking Neuron Networks (SNNs), also known as the third generation of neural networks, are inspired from natural computing in the brain and recent advances in neuroscience. SNNs can overcome the computational power of neural networks made of threshold or sigmoidal units. Recent advances on event-based devices along with their great power, considering the time factor, make SNNs a cutting-edge priority research objective. SNNs have been used mainly for classification problems, but their application to regression tasks remains challenging due to the complexity of training with continuous output data. In the literature we can find some first approximations in regression, specifically, for problems of a single variable of continuous values. This work deals with the analysis of the behavior of SNNs as predictors of multivariable continuous values. For this, a data set based on events has been generated from a bouncing ball and an event-based camera. The goal is to predict the next position of the ball over time.

Keywords: Regression · Spiking Neural Networks · Neuromorphic Software · DVS

1 Introduction

Wolfgang Maass [1] classified past and current Artificial Neural Networks ANNs research into three generations. First generation of ANNs is characterised by a single layer of units using a binary activation function and Hebb learning rule, allowing networks to simulate any Boolean function [2]. Second generation is organised in multilayer architectures, using non-linear activation functions, that make them representationally meaningful to stack more than one layer, and the existence of their derivatives makes it possible to use gradient-based optimization methods for training [3]. Second generation networks with one hidden layer, are universal approximators, that is, they can approximate any continuous, analog function arbitrarily well [4]. With recent advances in availability of large labelled data sets, computing power in the form of general purpose GPU computing, and advanced regularisation methods, Deep Learning (DL) [5] have emerged in the last decade. Although they have allowed us to make breakthrough progress in

many fields, they are biologically inaccurate and do not actually mimic the actual mechanisms of our brain's neurons.

In the third generation, SNNs include the concept of time into their operating model, the idea is that neurons in the SNN do not fire at each propagation cycle (as it happens with traditional neural networks), but rather they fire only when a membrane potential reaches a certain threshold. This temporal dependence makes the dynamics of this model closely resemble that observed in primate brain activity [6].

Besides the biologically motivated definition of SNNs, there is a more pragmatic application-oriented view coming from the field of neuromorphic engineering, where SNNs are often called event-based instead of spiking [7]. Event cameras produce asynchronous events for each pixel instead of intensity images. These events are generated by changes in brightness, obtaining a high time resolution of the order of microseconds, low power consumption and reduced bandwidth, by emitting only independent events instead of complete images. The resulting spatiotemporal event patterns can then be processed through networks of spiking neurons [8].

This processing exhibits interesting properties that make it particularly suitable for applications that require fast and efficient computation and where the timing information of the input/output signals is crucial to make predictions correctly[9]. In general, most of the works addressed so far with SNN and event cameras have focused mainly on multiclass classification problems, and only a few have addressed regression tasks. In [10], authors propose the use of the membrane potential as the estimated output instead of spikes when using SNN. However, there is currently no unanimity within the scientific community as to whether these values of electric potential could be good estimators of continuous values.

In this paper we follow, and extend, the mentioned approach based on the membrane potential and a single output regression task, to apply SNN for a regression task of two continuous output variables, the prediction of the position of a bouncing ball. We have selected two specific SNN models in order to assess their behaviour when membrane potential is taken into account in the learning process.

To address this task, different considerations must be taken into account about the spikes treatment both at the input of the SNNs as well as about the specific SNN models.

The remainder of this paper is organized as follows: Section 2 reviews recent literature surrounding SNN applications, spike coding and regression tasks. Section 3 describes the methodology we follow to generate the dataset, performance metrics adopted, the neuron models and the specific architectures based on these models. Section 4 details the results obtained through the experiments and finally we present in section 5 some concluding remarks and future research.

2 Related work

Although SNNs still lag behind DL networks in terms of their performance, the gap is vanishing on some tasks, while SNNs typically require much lower energy for the operation. SNNs have largely been applied to image classification on large datasets such as ImageNet[11][12], other works have been focused on object detection tasks [13], [14], pose estimation, action identification, etc. [15], [16], [17], [18], [19]. When computational resources or temporal information are crucial, SNNs could be the desired alternative. However most applications of SNNs are still limited to less complex datasets such as MNIST, N-MNIST, and N-Caltech101 [20], due to the complex dynamics of neurons and the non-differentiable nature of spike operations [21].

In regression tasks just a few research can be found based on event data or SNNs as learning models. In [22] they predicted a car steering angle using a Deep Learning approach specifically designed to work with the output of event sensors. The first, significant work, that considered SNNs as part of the learning architecture, was in [23], where authors dealt with the prediction of angular velocities of a rotating event camera in continuous time, they used a convolutional spiking neural network architecture with three, non-spiking, output neurons. More recently, [10] presented an interesting framework for nonlinear regression using SNNs. As they pointed out the use of SNN in regression it is determined by the treatment of events, not only at the input layers but the transformation into real numbers or continuous values of the output spikes.

At the input layers event data representations encode the event information related to a time-interval or temporal-window extracted from an event-stream.

In some works, events have been transformed into image-like representations compatible with natural images. The simplest way to use event data for supervised learning is to accumulate events pixel-wise over a period of time, either by counting them or by accumulating their polarities[22]. Other approaches include: 2D time surfaces or maps of most recent timestamps [24] or interpolated voxel grid [25]. This representation better preserves temporal information but requires more memory and more computations.

At the output layer, different spikes decoding strategies exist, based on rate decoding or latency decoding [26], however we follow the proposal of [10] where they designed a network topology based on different SNN layers and the output of the last spiking layer were transformed into a decoding layer taking the membrane potential of every time step as input and providing real numbers as output.

3 Methodology

3.1 Event-based Dataset

Events produced by a bouncing ball were recorded using a DVXplorer event camera [27], which provides a high dynamic range, high temporal resolution,

low latency, and low power consumption. This sensor is able to capture pixel-level brightness changes instead of standard intensity frames, consisting of a continuous flow of pixel events which represent the moving objects in the scene. At each pixel position $u_i = (x_i, y_i)$, the event-based camera produces events $e_i = (u_i, t_i, p_i)$ with polarity $p_i = (-1, 1)$ when the intensity changes above a threshold value at timestamp t_i [26].

Once the events have been recorded, we need to apply a representation method that converts asynchronous streams of events from event-based cameras to a sequence of sparse and expressive event frames [28]. Inspired by [22] the events recorded were binned over a time interval T (T=1000 microseconds) in a pixel-wise manner, obtaining 2D histograms of events. We used Tonic package [29] to transform events. The resolution of the camera were resized to 176×144 and time steps fixed to 100 miliseconds, providing a total of 166 x 100 frames.

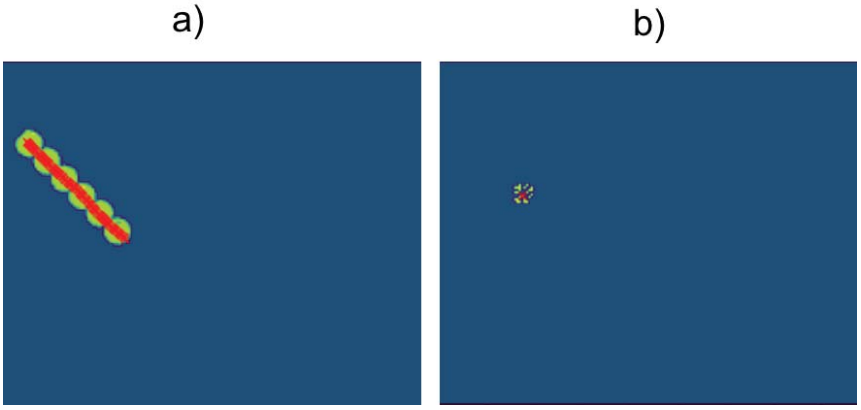

Fig. 1. Integrated events into 1000 microseconds bins, these events were captured by an event-based camera, the sensor were resized to 176x144 pixels. Each pixel (x,y) reports when it detects a relative change in the illumination intensity that is above or below a defined percentage of the previous intensity. In a) A 2D event frame (of size 176×144) accumulated over 100 time steps, red cross indicates the interpolated centers of the ball b) A 2D event frame (size 176x144) of a single time step and the interpolated center for this event set.

To obtain a labelled dataset, events must be associated with the exact position of the ball, so we applied the E2VID software [30] to generate a series of intensity frames from the event flow, in which the centers of the ball are calculated applying computer vision techniques. Once the position in each frame is obtained, a cubic interpolation is performed to calculate the position in each desired instant of time. Due to the nature of the data, they are asynchronous events with a temporal resolution of 1ms, so we obtained the position of the object at that resolution and averaged over the fixed time step. Fig. 1a) shows 100

accumulated frames and the interpolated centers of the ball. Fig. 1b) shows a single frame from the previous series of 100 binned frames and the corresponding center of the ball.

For learning purposes this dataset was split into two groups, the training set (70%) and the test set (30%).

3.2 Performance Metrics

To evaluate the performance of our network we used the root-mean-square error (RMSE), that measures the average difference between values predicted by a model and the actual values. It provides an estimation of how well the model is able to predict the target value (accuracy).

As a variance, RMSE can be interpreted as the standard deviation of the unexplained variance, and it has the useful property of being in the same units as the response variable. The lower the value of the Root Mean Squared Error, the better the model is.

The RMSE is particularly useful for comparing the generalization capacity of different regression models.

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{N}(target_i - predicted_i)^2}{N}} \tag{1}$$

3.3 Neuron Model

In this work we used the Leaky Integrated-and-Fire (LIF) model given its computational efficiency and capability of capturing the essential features of information processing in the nervous system [31].

The LIF model is represented by a RC circuit with a certain threshold. The state of a neuron is described in terms of its membrane potential, which is determined by the synaptic inputs and the injected current that the neuron receives. When the membrane potential reaches the fixed threshold, an action potential (spike) is generated and the voltage is reset[32][33] (see Fig. 2).

This neuron is modelled by the following equation:

$$\tau_m \frac{du}{dt} = -(U - U_r) + RI_{in}(t) \tag{2}$$

We refer to U as the membrane potential and to τ_m as the membrane time constant of the neuron, I_{in} is the input current and R the membrane resistance.

The standard LIF is a feed-forward neuron, then in order to take into account the changes in the recurrent connections and the dynamic strength of neuron connections over time, we also considered in our experiments the Recurrent LIF (RLIF) network.

RLIF assumes feedback connections where all spikes from a given layer are first weighted by a feedback layer before being passed to the input of all neurons[10]. This enables the network to use relationships along several time steps for the prediction of the current time step [33][26].

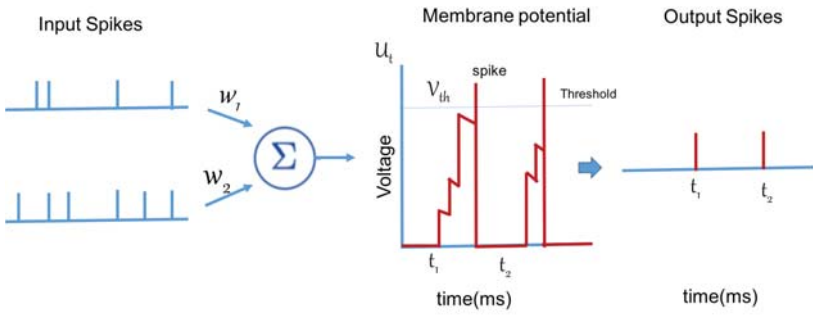

Fig. 2. The LIF model performs spatial and temporal integration of synaptic inputs,has a leaky membrane, generates a spike when the voltage reaches a certain threshold and goes refractory during the action potential

3.4 Neural Architecture

When using event-based data the key question in regression is how to interpret the binary spike information in the output layer to obtain real numbers. In[10] authors proposed training membrane potential to regress a single output variable, following this work we trained the membrane potential of LIF and RLIF-based SNNs consisting of two output units of continous values, in order to be able to predict the 2D position of a bouncing ball over time.

To this end, a three-layer SNN architecture was adopted and tested on the event-based dataset using LIF and RLIF principles. In every case the architecture consisted of 3 fully connected layers, the input layer of 176x144 units, two hidden layers of H and H/2 units and 2-units output layer. Figure 3 illustrates the general topoloy used in this work.

The inputs of the first, second and third layer are passed through the fully connected activation function before entering the LIF/RLIF neurons. Therefore, information flows in the form of accumulated event frames into the input layer, then is transformed into binary spikes in the spiking layers and taking back as real numbers in the output layer.

The SNNs tested in this paper was developed in snnTorch[34], a Python package for performing gradient-based learning with spiking neural networks, extending the capabilities of PyTorch[35].

Networks were trained using the fast sigmoid as surrogate gradient function. The loss function was the mean-squared error (MSE) between the output and the target and the optimization method was ADAM with a learning rate of 0.001. Exponential decay rates for the first and second moment estimates was 0.95.

We assessed the performance of SNN to investigate the following questions:

a) Are LIF and RLIF models capable of predicting continuous event-based values with a sufficient accuracy?

b) How can the number of hidden units influence the accuracy of the results?

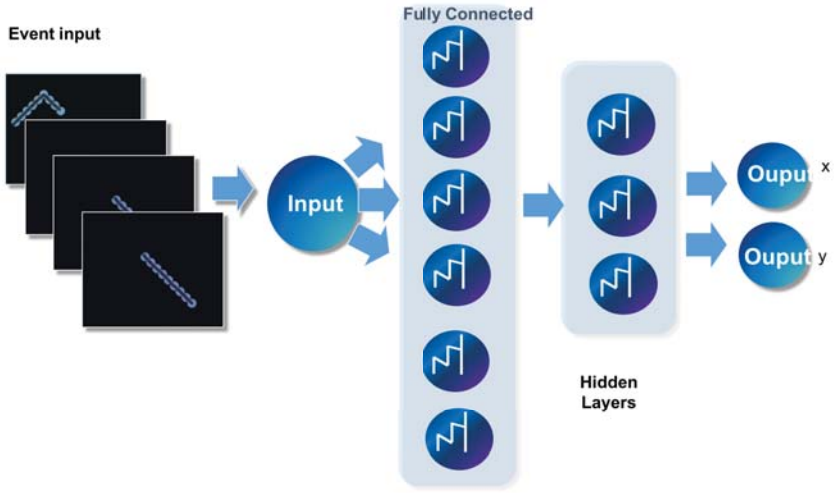

Fig. 3. LIF network scheme to illustrate the topology used in this work: a three-layer LIF network with H and H/2 hidden units at first and second hidden layers and 2 output units to predict the XY coordinates of the center position of the bouncing ball.

First, in order to check the predicting capability of LIF and RLIF models, we trained both, LIF and RLIF SNNs during 40 epochs and using 50 hidden units, we considered the continuous values obtained from the membrane potential of the SNN output units as final response of the network. The purpose was to study if membrane potential values could be good estimates of the multivariate continuous target values instead of using some kind of codification of the output spikes.

Once this issue was checked, we designed a more complete battery of experiments in order to assess the generalization capability of the different models and architectures and the influence of the number of hidden units.

We compared results between LIF and RLIF SNNs with three different number of hidden units H, namely, H=50, H=200 and H=500 for every first hidden layer, as well as the number of hidden units in the second layer was always assigned to half the number of neurons of the previous layer (H/2).

4 Results and Discussion

Every experiment was repeated 500 times and average values were calculated. We use box plots in order to visually show the distribution of numerical averaged RMSE and skewness by displaying the data quartiles (or percentiles) and averages.

For LIF based architectures of 50, 200 and 500 hidden units, figure 4 shows the RMSE error obtained after training and testing. From these results, it is

straightforward to observe that the more hidden units, the more accurate results. As the training progresses along the epochs all the networks are capable of improving the results. But always 500 hidden units provide a better accuracy, maybe if less complex networks were trained during more epochs we could get acceptable results with all the networks, but at the cost of increasing the computation time.

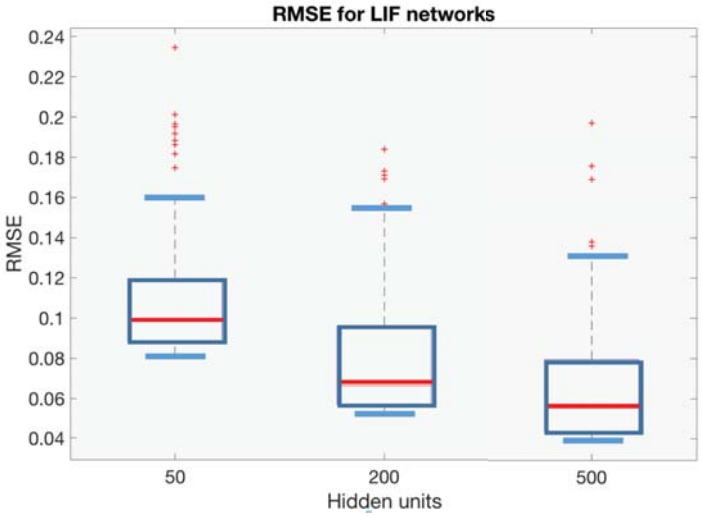

Fig. 4. Boxplot of RMSE test measurements for three different LIF SNNs.

For RLIF models the results show a different behaviour concerning the number of hidden units. In this case, the three architectures presented more similar averaged RMSE values as we can observed from the boxplots of figure 5.

We also show in figure 6 the line plots of the three averaged test RMSE measurements calculated over 100 epochs. On one hand, RMSE values for 50 and 500 hidden units are very similar during the first 20 epochs and then there exists a slight difference in favor of the 500 hidden units, since the RMSE measurement goes down during some more epochs to rise again from 27 epochs keeping variable during the following epochs.

The difference between 50 and 500 hidden units is not significant enough to opt for the most complex network. Thus, by analyzing the number of hidden units of the RLIF SNNs, although 50 hidden units could require more epochs to train the model, its simplicity against the 500 hidden units compensated the generalization capability, and did not suppose a significant increase in computation time. In general, it can be observed that 500 units did not improve the accuracy of the SNNs.

Table 1 reports the accuracy of each architecture in the form of the RMSE measurement, that allows us to quantitatively identify the models and the ar-

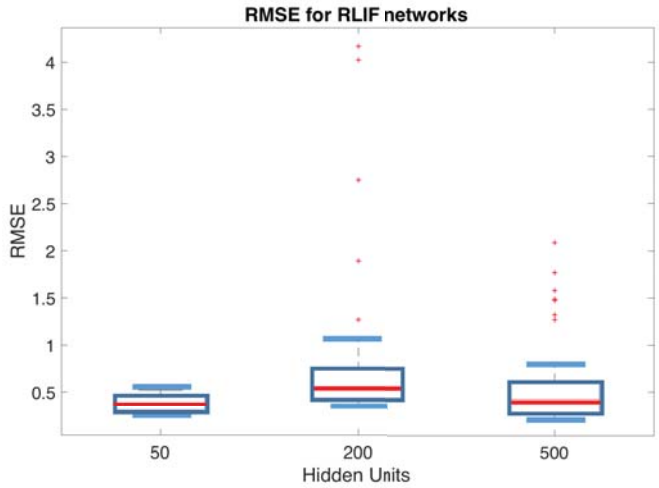

Fig. 5. Boxplot of RMSE test measurements for three different RLIF SNNs.

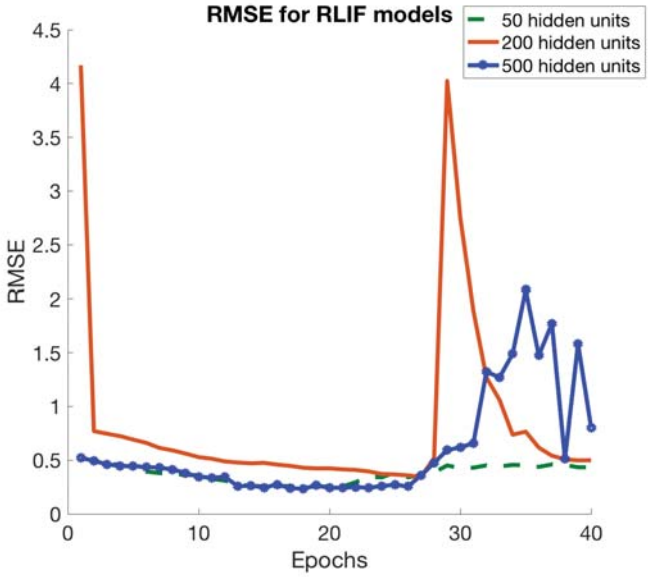

Fig. 6. Line plots of RMSE test measurements for three different RLIF SNNs.

chitectures that better fit the data. Last column indicates the number of epochs that each network reached the minimum test RMSE.

In all the cases we can observe that LIF models outperformed RLIF counterparts, which tell us that they could obtain a better generalization capability than RLIF based networks. Recurrent LIF networks enable the network to use relationships along several time steps for the prediction of the current time step, so it could be expected that these networks performed better than simple feedforward LIF models, but with this specific task, better results were obtained with non-recurrent models. Thus, in this regression task, a feedforward spiking neural network based on LIF model can predict relatively well the position of the ball, and the more complex network the more accurate results. However it is important to bear in mind that training neuromorphic data is expensive as it requires sequentially iterating through many time steps, and this cost is incremented as the complexity of the network.

Table 1. Generalization Error

Model	Hidden Units	RMSE	Epochs
LIF	50	0.0815	99
LIF	200	0.0518	98
LIF	500	0.0392	99
RLIF	50	0.2521	18
RLIF	200	0.3460	27
RLIF	500	0.2340	18

5 Conclusions and Future Work

The goal of this work has been to predict the position (XY coordinates) of a bouncing ball from 2D frames of events and using Spiking Neural Networks as predictors of these continuous values, using the membrane potential of LIF and Recursive LIF SNN models as the estimated outputs instead of the corresponding spikes. Results show how membrane potential on the output of SNNs models can be used as estimators of these continuous values with sufficient accuracy and encourage the research towards the comparison with other SNNs models and architectures. In addition we consider, as future work, the inclusion of convolutional layers to extract important features from the 2D event frames as well as we will investigate more complex datasets with different architectures of Deep SNNs for regression tasks, which will allow its application in robotic systems to object tracking.

6 Acknowledgements

F. M. Quintana would like to acknowledge the Spanish *Ministerio de Ciencia, Innovación y Universidades* for the support through FPU grant (FPU18/04321).

This work was also supported by the project NEMOVISION from the *Ministerio de Ciencia e Innovación y Agencia Estatal de Investigación*, PID2019-109465RB-I00/ MICIU/AEI /10.13039/501100011033.

References

1. Maass, W. Networks of spiking neurons: the third generation of neural net models. Neural networks, 10(9), 1659–1671, (1997).
2. McCulloch, W. S., Pitts, W. A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys, vol.5(4), pp.115-133 (1943).
3. Rumelhart, D. E., Mcclelland, J. L. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press, (1986).
4. Cybenko, G. Approximation by Superpositions of a Sigmoidal Function. Mathematics of Control, Signals and Systems, 2, 303-314, (1989).
5. LeCun, Y., Bengio, Y., Hinton, G. Deep learning. Nature 521, 436–444 (2015).
6. Gerstner, W., Kistler W.M., Naud R., Paninski L. Neuronal dynamics: From single neurons to networks and models of cognition. Cambridge University Press.(2014)
7. Human Brain Project Homepage, https://https://www.humanbrainproject.eu/en/. Last accessed 20 Mar 2023.
8. Hopkins, M., Pineda-García G., Bogdan P.A., Furber S.B. Spiking neural networks for computer vision. Interface Focus 8: 20180007.
9. Gallego G., Rebecq H., Scaramuzza D.. A Unifying Contrast Maximization Framework for Event Cameras, with Applications to Motion, Depth. IEEE/CVF Conf. on Comp Visi and Patt Recogn., 3867.(2018).
10. Henkes A., Eshraghian J.K. Spiking Neural Networks for Nonlinear Regression. IEEE Transactions on Neural Networks and Learning Sytems, October 26, (2022).
11. Rueckauer B, Lungu IA, Hu Y, Pfeiffer M, Liu SC. Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification. Front Neuroscience Dec 7;11:682.(2017).
12. Sengupta A., Ye Y., Wang R., Liu C., Roy K. Going Deeper in Spiking Neural Networks: VGG and Residual Architectures.Front. Neurosci.2019;13:95. doi:10.3389/fnins.2019.00095 (2019).
13. Kheradpisheh S.R., Ganjtabesh M., Thorpe S.J., Masquelier T. STDP-based spiking deep convolutional neural networks for object recognition.Neural Networks.99:56–67 (2018).
14. Zhou S., Chen Y., Li X., Sanyal A. Deep SCNN-Based Real-Time Object Detection for Self-Driving Vehicles Using LiDAR Temporal Data.IEEE Access.8:76903–76912 (2020).
15. Tavanaei A, Ghodrati M, Kheradpisheh SR, Masquelier T, Maida A. Deep learning in spiking neural networks. Neural Networks 111,47-63 (2019)
16. Mueggler E., Rebecq H., Gallego G., Delbruck T., Scaramuzza D. The Event-Camera Dataset and Simulator: Event-based Data for Pose Estimation, Visual Odometry, and SLAM. International Journal of Robotics Research, 36(2) 142-149, (2017).
17. Rebecq H., Horstschaefer T., and Scaramuzza D. Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization. In Proceedings of the British Machine Vision Conference 2017, BMVC 2017, BMVA Press, London, UK, (2017).

18. Kirkland, P., Di Caterina, G., Soraghan, J., Andreopoulos, Y., Matich, G. (2019). UAV Detection: A STDP Trained Deep Convolutional Spiking Neural Network Retina-Neuromorphic Approach. In: Artificial Neural Networks and Machine Learning – ICANN 2019: Theoretical Neural Computation. Lecture Notes in Computer Science(), vol 11727. Springer, Cham. (2019)
19. Orchard, G., Meyer C., Etienne-Cummings R., Posch C., Thakor N., Benosman R.HFirst: A Temporal Approach to Object Recognition. IEEE Transactions Pattern Analysis and Machine Intelligence, 37, 2028, (2015).
20. Orchard G., Jayawant A., Cohen G.K., Thakor N. Converting static image datasets to spiking neuromorphic datasets using saccades. Frontiers in Neuroscience 9:437 (2015).
21. Yamazaki K., Vo-Ho V-K, Bulsara D., Le N. Spiking Neural Networks and Their Applications: A Review. Brain Sci.12(7), 863 (2022).
22. Maqueda A. I. , Loquercio A. ,Gallego G. , García N., and Scaramuzza D. Event-based vision meets deep learning on steering prediction for self-driving cars. In Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition, 5419–5427, IEEE Computer Society, Salt Lake City, UT, USA, June (2018).
23. Gehrig M., Bam Shrestha S., Mouritzen D., Scaramuzza D. Event-Based Angular Velocity Regression with Spiking Networks. IEEE International Conference on Robotics and Automation (ICRA), Paris (2020).
24. Lagorce X., Orchard G., Galluppi F., Shi BE, Benosman RB. HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition. IEEE Trans Pattern Anal Mach Intell. 2017;39(7):1346-1359 (2017).
25. Shrestha B., Garrick Orchard G. SLAYER: Spike Layer Error Reassignment in Time. In Advances in Neural Information Processing Systems, pp. 1417-1426 (2018).
26. Eshraghian J. K., Ward M., Neftci E., Wang X., Lenz G.,Dwivedi G., Bennamoun M., Jeong D.S., Lu W.D. Training Spiking Neural Networks Using Lessons From Deep Learning. Neural and Evolutionary Computing (2022).
27. Inivation. Understanding the performance of neuromorphic event-based vision sensors, https://inivation.com/dvp/white-papers/ (2020)
28. Liu, Q., Pineda-García G., Stromatias E., Serrano-Gotarredona T., Furber S. B. Benchmarking spike-based visual recognition: a dataset and evaluation. Frontiers in Neurosciences.10:496 (2016)
29. Tonic 1.2.6, https://tonic.readthedocs.io/en/latest/. Last accessed 20 Mar 2023.
30. Rebecq, H., Ranftl R., Koltun, V. & Scaramuzza, D. High Speed and High Dynamic Range Video with an Event Camera. IEEE Trans. Pattern Anal. Mach. Intell. (T-PAMI) (2019).
31. Nunes J. D., Carvalho M., Carneiro D., Cardoso J. S. Spiking Neural Networks: A Survey. In IEEE Access, vol. 10, pp. 60738-60764, (2022)
32. Burkitt, A. N.. A review of the integrate-and-fire neuron model: I. homogeneous synaptic input. Biol. Cybern. 95, 1–19. doi: 10.1007/s00422-006-0068-6 (2006).
33. Wang Z., Zhang Y., Shi H., Cao L., Yan C., Xu G. Recurrent spiking neural network with dynamic presynaptic currents based on backpropagation. International Journal of Intelligent Systems, (37-3) 2242-2265 (2022).
34. Snntorch package. https://snntorch.readthedocs.io/en/latest/, last accessed 30/03/2023.
35. Pytorch webpage. https://pytorch.org, last accessed 20/03/2023.