SpikeBALL: Neuromorphic dataset for object tracking

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

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

Abstract. Most of widely used datasets are not suitable for Spiking Neural Networks (SNNs) due to the need to encode the static data into spike trains and then put them into the network. In addition, the majority of these datasets have been generated to classify objects and can not be used to solve object tracking problems. Therefore, we propose a new neuromorphic dataset, *SpikeBALL*, for object tracking that contributes to improve the development of the SNN algorithm for these type of problems.

Keywords: spiking neural networks · event cameras · object tracking · event-based dataset.

1 Introduction

During the last years, deep learning has reached a level of human-like performance in many areas [1, 2] owing to the combination of large scale datasets and increased computing power. Nevertheless, for gradient-based algorithms and the floating-point-based computation, artificial neural networks (ANNs) lack biological features and interpretability [3]. The current deep learning technology can be improved by blending computational neuroscience related knowledge and computer technology. The so-called third generation of artificial neural networks [4], Spiking Neural Networks (SNNs), simulate the human brain in terms of calculations and their representations, which shows strong biological interpretability. Neuromorphic engineering is an emerging paradigm in which elements of a computer are modeled after the human brain and nervous system. Mathematical models and their implementation in physical circuits are required for that purpose. Neuromorphic architectures are most often modelled from neurons and synapses. Neurons use electronic and chemical impulses to send information between different regions of the brain and the rest of the nervous system. Neurons use synapses to connect to one another and both are far more flexible, adaptable and energy-saving information processors than traditional computer systems.
Due to the natural characteristics of SNNs, their hardware implementation (analog or digital) generate advantages like real time processing and low consumption

[5]. SNNs operate using spikes, which are discrete events that take place at points in time, rather than continuous values. The occurrence of a spike is determined by differential equations that represent various biological processes, the most important of which is the membrane potential of the neuron. Essentially, once a neuron reaches a certain potential, it spikes, and the potential of that neuron is reset.

In this work, we propose a new neuromorphic dataset, *SpikeBALL*, for object tracking that contributes to improve the development of the SNN algorithm for this type of problem in which, given the initialization of a specific objective (position of the object (x, y)), the trajectory of the objective will be followed in the recording set obtained by the event camera. The dataset is formed by trajectories of a ball generated by the players of a table football and belong to a single ball The dataset is made up of trajectories of a ball generated by the players of a table football and belong to a single ball hit by the players, following the rules of this game. These trajectories have been obtained by the event camera and their objective is to train a SNN to follow the ball and predict its trajectory.

For that purpose, the background of this problem is presented in Section 2. The methodology that has been followed to generate the dataset is presented in Section 3. Then, we describe the dataset in Section 4. Finally, the conclusions and future works are shown in Section 5.

2 Background

In recent years, many researchers have shown much interest in object tracking in video sequences [6]. In the field of image processing and computer vision, detecting the objects in the video and tracking its motion to identify its characteristics has been emerging as a demanding research area. Object tracking have been developed very fast in the past few years, most of these works have been developed using Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), however, few of them have used SNN to solve this type of problem [17]. Object tracking is a nontrivial problem in computer vision, and is widely used in sports events broadcasting, robotic, security monitoring and other fields. The use of Siamese networks [7] and Transformers have achieved to become very mature the object tracking with traditional cameras. However, when the objects are moving under extreme conditions of high speed and high dynamic range, traditional cameras have difficulty capturing their movements.

Recent developments in neuromorphic computing systems have focused on the creation of new hardware based on biological characteristics. An example of such hardware are event cameras, such as the Dynamic Vision Sensor (DVS), in which pixels of the DVS work independently (i.e., asynchronously). Each pixel announces when it discloses a relative change in the illumination intensity that is above or below a defined intensity threshold. A further advantage of this asynchronous approach is its enhanced dynamic range, since the pixel sensitivity does not have to be set globally in the event camera. In such a way, the unnecessary

static information, such as background landscapes, is not recorded, only the dynamic information is registered. Consequently, event cameras are robust and accurate motion detectors and automatically filter out any temporarily redundant information [8]. This makes them extremely useful for scenes with motion like high-speed counting, or driving safety systems.

The currently widely used datasets, such as ImageNet[9] and COCO[10], have contributed significantly to the success of deep learning. Nevertheless, these type of datasets are not suitable for SNNs due to the need to encode the static data into spike trains and then put them into the network [11]. SNNs need datasets generated by neuromorphic camera DVS avoiding the missing information and can be fair to compare with the artificial neural networks. Researchers have proposed many neuromorphic datasets using DVS such as N-MNIST[12] or DVS-CIFAR10[13] which are traditional classification datasets that follow predetermined or random trajectories of motion. On the other hand, other datasets has been obtained recording activities in natural environments using neuromorphic cameras, such as DVS-Gesture[14] and N-Cars[15]. However, the majority of these datasets have been generated to classify object and can not be used to solve object tracking problems. For this reason, with the creation of this dataset, we intend to contribute to the development of neuromorphic databases in which regression problems can be solved.

3 Methodology

In this work, a methodology have been developed to keep track of a ball inside a table football. The procedure that has been followed begins with the automated capture data. For that purpose, we use the DVS acquisition platform to shoot videos that are played on a monitor, and use the DV software[16] to collect the data automatically. The second stage consists of post-processing data in which the noisy data are removed. Finally, the data is labeled. Fig. 1 shows the entire construction process of the dataset.

3.1 Automated capture data

The acquisition device used for generating the dataset have been DVXplorer camera of Innovation Company[16] which allows videos with a 640x480 resolution. DVS camera prevents external light changes from interfering with the experimental data collection. To display the trajectory of the ball, it has been necessary to design a bracket to hang the camera from the ceiling, this bracket has been created on a 3D printer. Fig. 2 shows the equipped laboratory with the football table and the position of the event camera, hanging from the ceiling in the vertical line, perpendicular to the table, in order to capture the trajectories of the ball.

In the experiment, the DV software is used to process the captured event data. When a player kicks the ball, the automatic capture process described in figure 1*A* is activated. A video is obtained for each hitting the ball. This process is finished when no player kicks the ball.

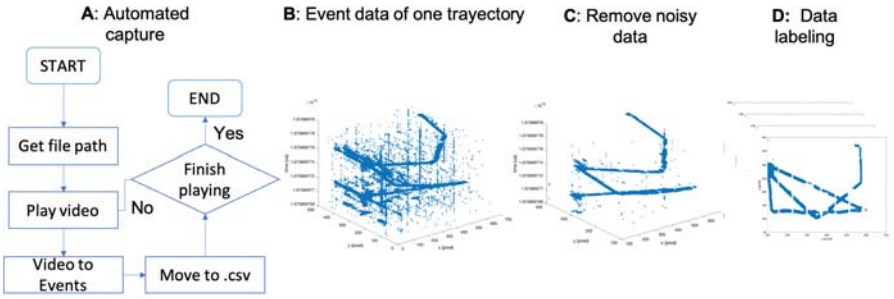

Fig. 1. Complete process of data generation. Phase *A* is the capture stage, including the building of the equipment environment, and the recording with the DV software. Phase *B* is the data acquisition as event data format. Phase *C* performs post-processing removing noisy data. And Phase *D* carry out the data labeling.

Fig. 2. The equipped laboratory to capture the trajectories of the ball with the event camera hanging from the ceiling.

3.2 Post-processing of data

The obtained videos contain information about the track of the ball, but also about the rods which have figures attached. For this reason, an algorithm has been developed in order to remove data that do not take in part of track of the ball. This algorithm has been designed exclusively for these dataset in order to obtain the cleanest tracks possible. It is shown, broad terms, in Algorithm 1 and it consists of determining a region which allowing computation of points that are part of the track.

Algorithm 1 Remove noisy data

```
Determine a optimal size region
for Every point i in trajectory do
    Calculate the region for i
    Save the points inside the region
    Add these points to the set of points belonging to the trajectory of the ball
end for
```

3.3 Data labeling

Data labeling is carried out from the cleaned data. The track of the ball is divided into time frames of 10ms. The centre (cx, cy) of all the points that belong to a time frame t is calculated and (cx, cy) is the label for all points of the time frame t. This process is repeated for each time frame that have been divided the track of the ball. It is shown in Algorithm 2.

Algorithm 2 Data labeling

```
Divide the track in time frames of 10ms
for Every time frame t do
    Select the points that belong to t
    Calculate the centre (cx, cy) of these points
    Tag the points with the centre (cx, cy)
end for
```

4 Data base description

In order to store the dataset, we save the event data with the form of (x, y, t, p), where the first two items x, y are the pixel coordinates of the event, the third item t is the timestamp of the event, and the fourth item p is the polarity with value 1 and 0 indicating the increase or decrease of brightness separately. The two

polarities are represented by two channels, and the pixels without events are filled with 0. In addition, the labels corresponding to the centers (cx, cy), generated as mentioned in section 3.3 of this work, are stored. For each trajectory a *.csv* file is saved with the events write in previous format. The dataset is comprised of 10 different trajectories with between 80,000 and 100,000 events each one. One trajectory is shown in Fig. 3, in blue dots events positions (x, y) and green stars represent the target of these events.

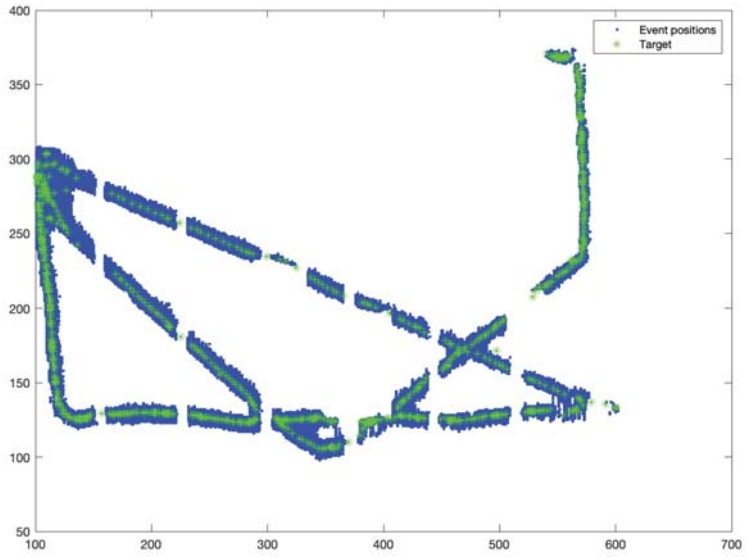

Fig. 3. One of the ten trajectories that are part of dataset. It comprises 83,580 events that are plotted in (x, y) format, where x, y are the pixel coordinates of the event in blue dots. Green stars are the targets of these events.

5 Preliminary results

5.1 Neural Arquitecture

With the intention of testing this dataset, *SpikeBALL*, a network has been used that avail of both the benefits of CNN and SNN.

On one hand, CNNs offer several distinct advantages that make them highly effective for computer vision tasks.These advantages include:

1. Local Connectivity and Shared Weights: CNNs leverage the concept of local connectivity, where neurons are connected to a small receptive field, resembling the receptive fields of neurons in the visual cortex. This local connectivity allows CNNs to capture local patterns and spatial dependencies efficiently. Additionally, CNNs employ weight sharing, where the same set of weights is shared across different spatial locations, enhancing parameter efficiency and generalization capability [18].
2. Translation Invariance: CNNs exhibit translation invariance, meaning they can detect and recognize patterns regardless of their location within an input image. This property is achieved through the combination of shared weights and convolutional operations, enabling CNNs to learn spatial hierarchies of features that are robust to translation [18].
3. Hierarchical Feature Extraction: CNNs automatically learn hierarchical representations of features from raw input data. Multiple convolutional layers allow CNNs to learn low-level features (e.g., edges, textures) in early layers and progressively extract higher-level features (e.g., shapes, objects) in deeper layers. This hierarchical feature extraction enables CNNs to capture complex patterns and learn rich representations [18].
4. Parameter Sharing and Model Size: CNNs significantly reduce the number of parameters compared to fully connected networks through weight sharing. By reusing the same set of weights across different spatial locations, CNNs achieve parameter efficiency, making them easier to train and less prone to overfitting, particularly with limited training data[18].
5. Spatial Invariance and Robustness: CNNs exhibit spatial invariance, enabling them to recognize patterns even with slight transformations, such as rotations or scale changes. This spatial invariance and robustness make CNNs well-suited for tasks where precise spatial location or scale is not critical, such as object recognition and image classification [18].

These advantages have contributed to the remarkable success of CNNs in various computer vision tasks, including image classification, object detection, semantic segmentation, and image generation.

On the other hand, SNNs offer several advantages over traditional neural networks. Firstly, SNNs capture the temporal dynamics of neural information processing by encoding information in the timing and order of spikes [4]. This temporal coding enables SNNs to represent time-dependent patterns and process dynamic inputs. Secondly, SNNs exhibit high energy efficiency due to their sparse and event-driven computations [23]. By generating spikes only when necessary, SNNs reduce overall power consumption and are well-suited for resource-constrained environments or low-power devices. Furthermore, SNNs have enhanced biological plausibility as they incorporate concepts such as refractory periods, spike timing, and synaptic plasticity mechanisms like Spike-Time-Dependent Plasticity (STDP) [4]. This biological fidelity makes SNNs valuable for studying and understanding neural information processing in biological systems. SNNs also demonstrate robustness to noise through their ability to filter out noisy input and focus on relevant spike timings [24]. This feature enables

SNNs to perform well in noisy or uncertain environments, enhancing their reliability in practical applications. In addition, SNNs operate in an event-driven manner, generating spikes only when significant changes occur in the input . This event-driven processing leads to efficient and real-time computations, as computational resources are allocated only when required.

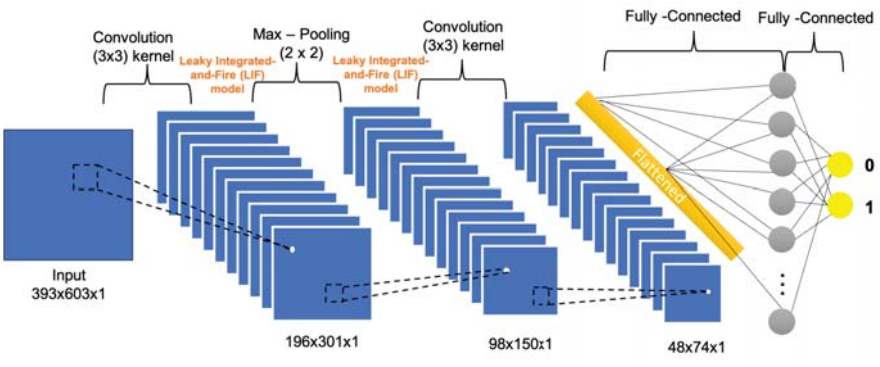

Fig. 4. Network architecture used that includes a CNN and an SNN.The SSN is implemented using a Leaky Integrate-and-Fire (LIF) model.

Fig. 4 shows the architecture of the network used to validate the set of data proposed in this work. The SSN is implemented using a Leaky Integrate-and-Fire (LIF) model. LIF model is a commonly used neuron model in computational neuroscience and neural network simulations. It provides a simplified representation of the behavior of biological neurons, capturing essential dynamics while maintaining computational efficiency [25]. In the LIF model, the membrane potential of the neuron is modeled as an electrical potential that evolves over time. It integrates incoming synaptic inputs and generates output spikes based on a threshold mechanism. Notably, the LIF model incorporates a leak term, which accounts for the gradual decay of the membrane potential over time. The LIF model is computationally efficient and allows for analytical analysis, making it a valuable tool for investigating neural dynamics and simulating SNN. Although the LIF model simplifies the intricate dynamics of biological neurons, it serves as a foundational framework for understanding basic principles of neural information processing.

5.2 Performance Metrics

We propose root-mean-square error (RMSE) for evaluating the accuracy of the predictive model. It measures the average magnitude of the differences between the predicted values and the actual values. The RMSE metric provides a single value that represents the typical error or deviation of the predicted values from

the actual values. It is especially useful in regression tasks, where the goal is to estimate a continuous target variable.

A lower RMSE indicates that the predicted values are closer to the actual values, implying higher accuracy and better model performance. On the other hand, a higher RMSE suggests larger errors between the predicted and actual values, indicating lower accuracy and poorer model performance.

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

5.3 Results and discussion

The process has been repeated 300 times and each time 60 epochs have been performed. Fig. 5 shows the RMSE calculated for each epoch. It can be seen that after 20 epochs the error stabilizes at around 1%.

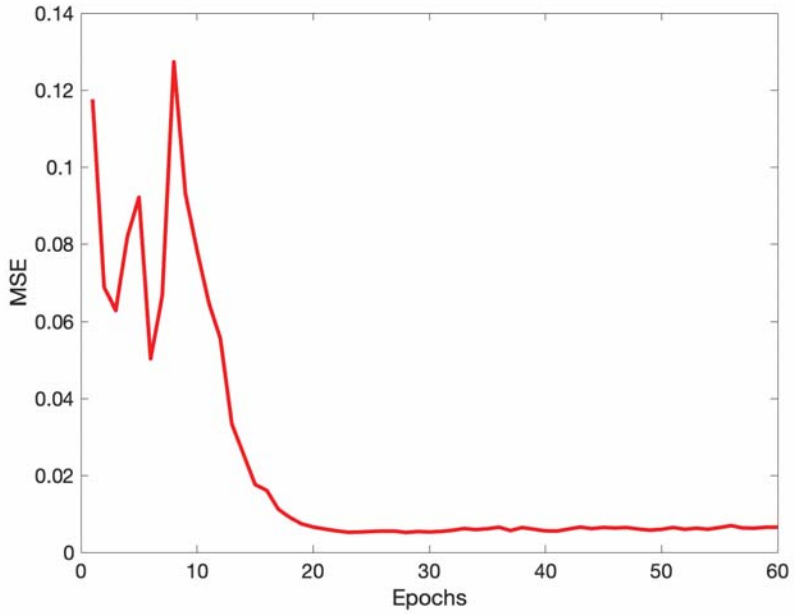

Fig. 5. RMSE depending on the epochs. After 20 epochs the error stabilizes at around 1%

To qualitatively measure how good the prediction is, Figure 6 shows one of the ten trajectories in which the positions of the events appear with blue dots, the centers calculated with the previously explained methodology with green stars,

and the network prediction with red stars. It can be seen that the prediction generates values very close to the calculated centers, which corroborates the RMSE values obtained.

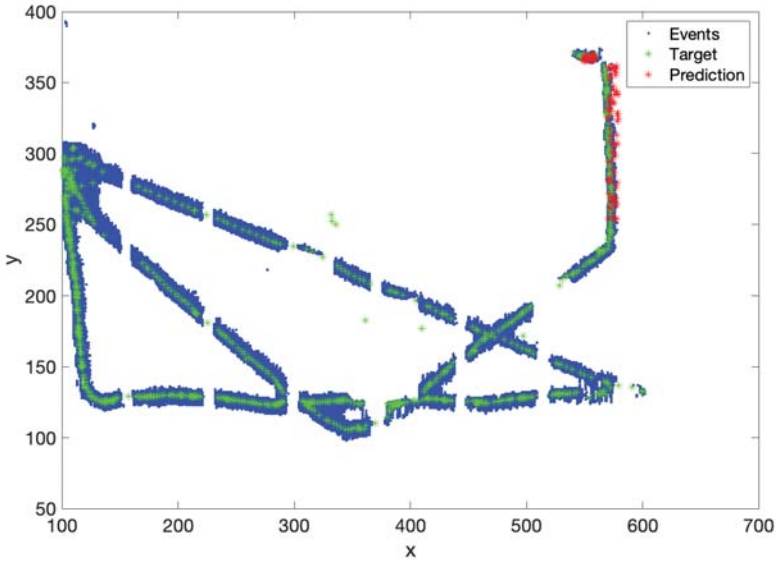

Fig. 6. One of the ten trajectories in which the positions of the events appear with blue dots, the centers calculated with the previously explained methodology with green stars, and the network prediction with red stars.

6 Conclusions and future works

In this work, we propose a new neuromorphic dataset, *SpikeBALL*, for object tracking that contributes to improve the development of the SNN algorithm for these type of problems. In this way, a methodology has been developed in order to obtain a huge dataset to determine the trajectory of the ball in a table football. The procedure that has been followed begins with the automated capture data, post-processing data and, finally, the data labeling. Following this methodology, a dataset, comprised of 10 different trajectories with between 80,000 and 100,000 events each one, has been generated. Thanks to developed methodology the dataset is highly scalable, being able to generate a large volume of data in a short time.

In order to test the effectiveness of *SpikeBall* and the potential to provide new challenges for training SNN algorithms, experiments have been conducted

on an architecture that combines the advantages of a CNN and an SNN.The results obtained show that *SpikeBall* can be useful to improve the prediction of SNNs in regression problems.

As future work, we think it would be good to use different architectures other than the one proposed together with classical regression methods to compare the results.

Acknowledgements

Fernando 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. Hirschberg, J., Manning, C. D. Advances in natural language processing. Science **349**, 261–266 (2015)
2. Noda, K., Yamaguchi, Y., Nakadai, K., Okuno, H. G., Ogata, T. Audio-visual speech recognition using deep learning. Applied Intelligence **42**, 722–737 (2015)
3. Rumelhart, D. E., Hinton, G. E., Williams, R. J. Learning representations by back-propagating errors. Nature **323**, 533–536 (1986)
4. Maass, W. Networks of spiking neurons: the third generation of neural network models. Neural networks **10**, 1659–1671 (1997)
5. Shen, G., Zhao, D., Zeng, Y. Backpropagation with biologically plausible spatiotemporal adjustment for training deep spiking neural networks. Patterns **100522** (2022)
6. Barga, D., Thounaojam, D.M. A survey on moving object tracking in video. International Journal on Information Theory **3** (2014)
7. Zhang, Z., Liu, Y., Wang, X., Li, B., and Hu, W. Learn to match: Automatic matching network design for visual tracking. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (IEEE), 13339–1334 (2021)
8. Inivation. Understanding the performance of neuromorphic event-based vision sensors, https://inivation.com/dvp/white-papers/ (2020)
9. Deng, J. et al. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition (IEEE), 248–255 (2009)
10. Lin T.-Y. et al. Microsoft coco: Common objects in context. In: European conference on computer vision (Springer), 740–755 (2014)
11. Zhang, T. et al. Self-backpropagation of synaptic modifications elevates the efficiency of spiking and artificial neural networks. Science Advances **7**, eabh0146 (2021)
12. 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)
13. Li, H., Liu, H., Ji, X., Li, G., Shi, L. Cifar10-dvs: an event-stream dataset for object classification. Frontiers in neuroscience **11**, 309 (2017)

14. Amir, A. et al. A low power, fully event-based gesture recognition system. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (IEEE), 7243–7252 (2017)
15. Sironi, A., Brambilla, M., Bourdis, N., Lagorce, X., Benosman, R. Hats: Histograms of averaged time surfaces for robust event- based object classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1731–1740 (2018)
16. Inivation Hompage, https://inivation.com. Last accessed 20 Mar 2023
17. Yongqiang, C., Yang, C., Deepak, K. Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition. International Journal of Computer Vision **113**, 54–66 (2015)
18. LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444
19. Simonyan, K., and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
20. Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25, 1097-1105
21. Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3431-3440.
22. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., and Rabinovich, A. (2015). Going deeper with convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1-9.
23. Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., and Jackson, B. L. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. Science, 345(6197), 668-673.
24. Sengupta, A., Ye, Y., Wang, R., and Liu, Y. (2019). Going deeper in spiking neural networks: VGG and residual architectures. Frontiers in Neuroscience, 13, 95.
25. Izhikevich, E. M. (2003). Simple model of spiking neurons. IEEE Transactions on Neural Networks, 14(6), 1569-1572.