# Fall Detection with Event-Based Data: A Case Study

Xueyi Wang[1(✉)], Nicoletta Risi[1], Estefanía Talavera[2], Elisabetta Chicca[1], Dimka Karastoyanova[1], and George Azzopardi[1]

[1] University of Groningen, Nijenborgh, 9742 AG Groningen, The Netherlands
{xueyi.wang,n.risi,e.chicca,d.karastoyanova,g.azzopardi}@rug.nl
[2] University of Twente, Drienerlolaan, 7522 NB Enschede, The Netherlands
e.talaveramartinez@utwente.nl

**Abstract.** Fall detection systems are relevant in our aging society aiming to support efforts towards reducing the impact of accidental falls. However, current solutions lack the ability to combine low-power consumption, privacy protection, low latency response, and low payload. In this work, we address this gap through a comparative analysis of the trade-off between effectiveness and energy consumption by comparing a Recurrent Spiking Neural Network (RSNN) with a Long Short-Term Memory (LSTM) and a Convolutional Neural Network (CNN). By leveraging two pre-existing RGB datasets and an event-camera simulator, we generated event data by converting intensity frames into event streams. Thus, we could harness the salient features of event-based data and analyze their benefits when combined with RSNNs and LSTMs. The compared approaches are evaluated on two data sets collected from a single subject; one from a camera attached to the neck (N-data) and the other one attached to the waist (W-data). Each data set contains 469 video samples, of which 213 are four types of fall examples, and the rest are nine types of non-fall daily activities. Compared to the CNN, which operates on the high-resolution RGB frames, the RSNN requires 200× less trainable parameters. However, the CNN outperforms the RSNN by 23.7 and 17.1% points for W- and N-data, respectively. Compared to the LSTM, which operates on event-based input, the RSNN requires 5× less trainable parameters and 2000× less MAC operations while exhibiting a 1.9 and 8.7% points decrease in accuracy for W- and N-data, respectively. Overall, our results show that the event-based data preserves enough information to detect falls. Our work paves the way to the realization of high-energy efficient fall detection systems.

**Keywords:** Fall detection · Wearable cameras · Event-based · Deep learning · RSNN · CNN · LSTM

N. Tsapatsoulis et al. (Eds.): CAIP 2023, LNCS 14185, pp. 33–42, 2023.
https://doi.org/10.1007/978-3-031-44240-7_4

## 1 Introduction

Elderly individuals often experience reduced control over their bodies, weaker bones, and longer recovery times in case of injuries, which increase the risk, severity, and impact of falls [1]. The act of falling poses a significant threat to the elderly population as it is considered a factor leading to a decrease in autonomy, fatalities, and harm [2]. These incidents can result in high healthcare costs and other related expenses, which can place a strain on both individuals and healthcare systems [3]. There is, therefore, a growing interest in the development of more effective, low-power, and privacy-friendly fall detection systems [4].

The aforementioned factors serve as our motivation to develop a fall detection system that can be embedded in Internet of Things (IoT) or edge computing with low energy consumption, privacy protection, and low-latency computation using minimal computational resources. In pursuit of this objective, the current study introduces a proof-of-concept to detect falls on the edge with IoT devices. This is achieved by interfacing the output of event cameras – generated via the conversion of an existing dataset with RGB video clips – with a Spiking Neural Network (SNN), which requires relatively few parameters and multiply-accumulate (MAC) operations.

Unlike traditional frame-based cameras, event-based cameras have independent pixels which respond only to changes in brightness over time [5]. The pixels operate asynchronously and report local brightness changes at the time of their occurrence. This approach has significant advantages, namely low energy consumption, high temporal resolution, low-latency event streams, and high dynamic range.

This work has two main contributions. Firstly, we provide a comparative analysis between a standard CNN operating on RGB frames along with a RSNN and a LSTM model trained on simulated event-based data. Secondly, we investigate the balance between performance and efficiency in relation to model complexity and energy consumption.

## 2 Related Works

In the context of fall detection, the types of sensors that have been investigated can be broadly classified into four distinct types: wearable, fixed visual, ambient sensors, and sensor fusion [6]. Wearable sensors are popular options due to their portability and ability to collect data without location limitations. Moreover, they can leverage the physiological variations of the human body. Fixed visual sensors are particularly useful as they consist of simplified hardware with good image quality and notable reliability. Different types of fixed visual sensors, including RGB cameras and RGB-D depth cameras, have been investigated in this regard [7]. The integration of wearable sensors and visual sensors has led to the emergence of wearable cameras as a promising alternative for fall detection. The fusion of various sensors can enhance the resilience of fall detection systems [6].

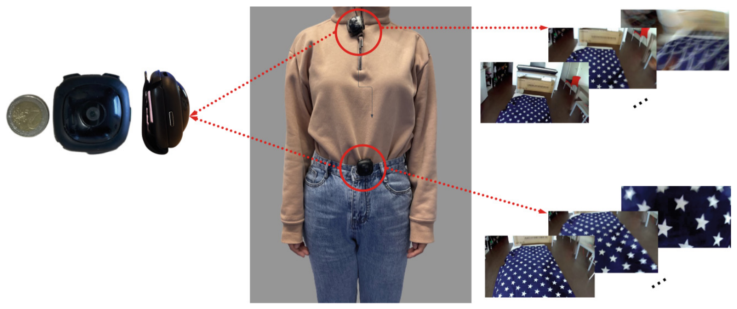

**Fig. 1.** Mounting of two cameras to the neck and waist of the participant. Left: front and profile photos of the camera compared to the size of a coin. Right: frame sequences collected by the wearable cameras from the two perspectives.

Over the past decade, event-based cameras have gained increasing popularity due to the advantages mentioned above. Also, event-based sensing has been employed to investigate action recognition on third-person view datasets, reflecting the increasing interest in this sensing paradigm [8]. In a recent study [9], the authors compared the performance of a CNN combined with a LSTM architecture on conventional gray-level frames with corresponding simulated event-based data with respect to human action recognition. Their results show the plausibility of using simulated event-based data to classify four different activities.

However, none of the studies mentioned above provide a fall detection solution, which is portable, low-power, and low-latency. We address these scientific challenges by investigating an event-based approach. By removing the need for cloud computing, our event-based method can provide enhanced privacy, speed, and security by processing data directly on the edge.

## 3 Methods

### 3.1 The Data Set

We introduce two new public datasets of 469 RGB video samples each, i.e., more than previous datasets [10,11], and the corresponding simulated event-based data. Two small and light-weight wearable cameras of the same type, (measuring $420 \times 420 \times 200$ mm and 25 g), were used for data collection, one attached to the neck and the other to the waist, Fig. 1. The two cameras were used to capture the events at the same time. In this respect, our dataset consists of two subsets of 469 samples each. For simplicity, the two categories will be referred to as *W-data* and *N-data*. Video recordings include 13 daily activities categorized into four types of *falls* (front falls, back falls, downside falls, and lateral falls), and nine *non-fall events* (lying, rising, sitting down, bending, stumbling, walking, standing, squatting, and sitting static). The samples were collected indoors and outdoors over two days, resulting in 213 falls and 256 non-falls per camera. The data was collected with a resolution of 1080p and a frame rate of 30 fps. Hereafter, our comparative analysis focuses on the binary classification task, i.e., on discriminating samples between *falls* and *non-falls*.

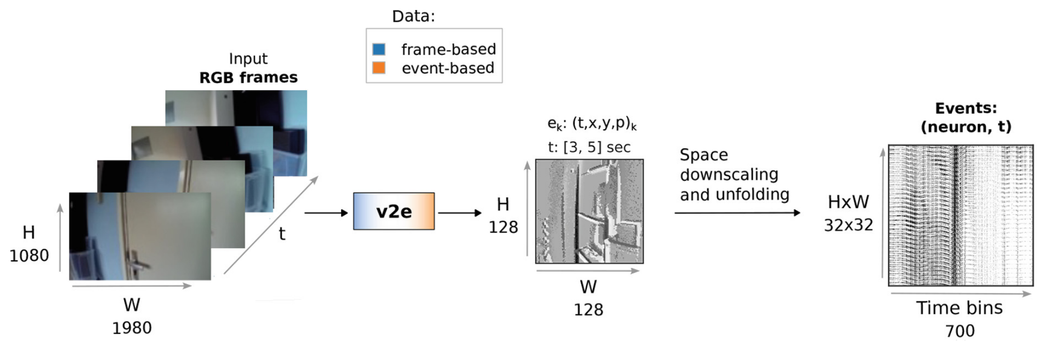

**Fig. 2.** Pipeline workflow with an example of a fall event. From RGB frames ($1980 \times 1080$ pixels) to streams of events $e_k = (t, x, y, p)_k$, depicted as time surfaces, via the event-camera simulator *v2e*. The final binary representation is achieved via space downscaling and unfolding, with non-zero values (dark) of downscaled pixels ($32 \times 32$) representing the occurrence of at least one event in the corresponding time bin.

**Data Splitting:** We use the same data split across the three models under investigation, CNN, LSTM, RSNN: 15% of the data is randomly selected as the test set, and the remaining data is used to generate 10 randomly selected train/validation splits with a ratio of 85:15.

**From Videos to Events (*v2e*):** The *v2e* simulator [12] was used to generate simulated event-based data for our experiments[1]. The tool *v2e* can produce highly realistic synthetic events from normal RGB frames. As event-camera model, we chose the Dynamic Vision Sensor (DVS)[2], with a resolution of $128 \times 128$ pixels. Figure 2 shows examples of original RGB frames and the corresponding event-based data for one sample of our dataset. The event data format consists of the 4-tuple $e_k = (t, x, y, p)_k$, where $t_k$, $(x_k, y_k)$ and $p_k$ refer to the time step, spatial coordinates, and polarity, respectively. An example of this data format is depicted in Fig. 2 as a time surface [5], with darker pixels indicating more recent time. Both frame- and event-based data formats are considered in our study, and processed with two pipelines; one processing the event time series (Sect. 3.2 A), and the other processing the RGB frame-based data (Sect. 3.2 B).

**Event-Based Data Pre-processing:** Before processing the event streams, the pixel array's spatial resolution was downscaled to $32 \times 32$ pixels to reduce the number of network parameters and computational resources needed for training the RSNN and LSTM. Despite the event-camera output's asynchronous nature, the models' simulation on a CPU requires defining a time binning step for the

[1] As a proof-of-concept study, we chose the conversion parameters for the event-camera model with no background noise, i.e., thresh = 0.2, sigma = 0.02, cutoff_hz = 0, leak_rate_hz = 0, shot_noise_rate_hz = 0.

[2] This choice was driven by ongoing research, which aims to compare current results with event-camera recordings using an embedded DVS (eDVS) [13].

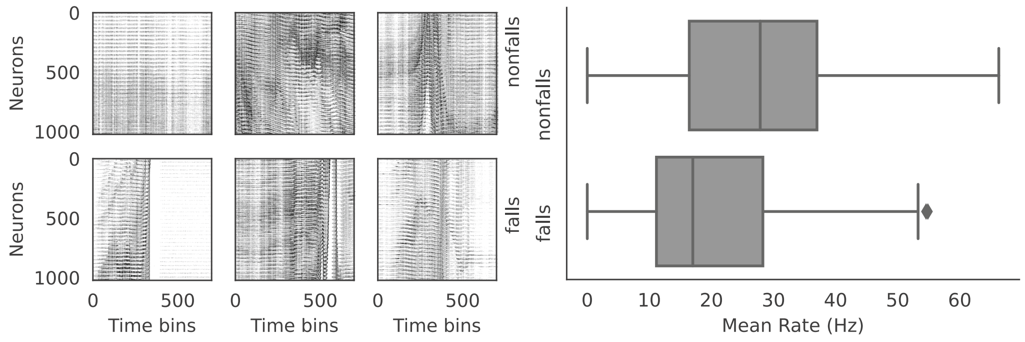

**Fig. 3.** (Left) Event-based data for 3 *non-fall* and 3 *fall* samples. (Right) Per-class distribution of pixels' mean firing rates across all samples in the dataset.

input event stream, which was set to $dt = 10$ms. To accommodate different video clip lengths, we extracted for each sample the time of maximum instantaneous firing rate $T^*$. Then, we cropped each video clip to a time window $\Delta_T = 7$sec, which was centered around the time point $t^*$ drawn from a uniform distribution $p(t) = 1/(2\Delta)$, with $t \in [T^* - \Delta, T^* + \Delta]$ and $\Delta = 500$ms. The polarity information was discarded, and the spatial information was collapsed to a single dimension, where $x$ and $y$ event coordinates are mapped to a single index. This dimension corresponds to the number of input neurons $\mathrm{n}_{\mathrm{neurons},1}$ of the RSNN (and input nodes of the LSTM). The resulting event-based representation has dimensions: $\mathrm{n}_{\mathrm{neurons},1} \times (10^3 \Delta_T)/dt$, Fig. 2. Figure 3 displays three randomly selected samples for each of the two classes, *fall* and *non-fall*.

To assess whether the simulated event-based data preserves the relevant information for the problem at hand, we first quantified the performance of a standard classifier, namely a linear Support-Vector Machine (SVM), operating on "time-collapsed data", i.e., when the temporal information of the input time series is removed. In this representation, each video sample is represented by a 1-dimensional vector, where each element is the sum of all events at the corresponding location. Results are illustrated and discussed in Sect. 4.

## 3.2 Fall Detection Approach

**A) Classifiers for the Event-Based Data.** This section describes the RSNN and the LSTM models used to classify the event time series.

***Events + RSNN:*** As the natural interface of event-based sensing is event-based processing, we assessed the performance of an RSNN for fall detection.

To this end, we adapted the RSNN model from [14] and trained it using Backpropagation Through Time (BPTT) with surrogate gradients [15]. Specifically, the network consists of three layers current-based (CUBA) Leaky Integrate and Firing (LIF) neurons. Compared to the LIF model, the CUBA LIF integrates the input spikes into a current variable prior to generating the membrane potential [16]. The input layer has one-to-one connections with the $32 \times 32$ down-scaled pixel array generated by *v2e*, Fig. 2. The two downstream layers are fully connected with plastic synapses, and with the output layer consisting of two CUBA LIF neurons, which encode the network prediction with one-hot encoding. By leveraging the approximation of the spiking non-linearity with a differentiable function, surrogate gradients were computed using PyTorch's differentiation. The network was trained by minimizing the Negative Log-Likelihood Loss ($L_{nll}$). For each input $\mu$, the network predicted probability $p_i$ of class $i$ was computed as the softmax of the maximum membrane potential of the readout units, $U_i[t]$ for $i \in [0, 1]$, measured across the time window $\varDelta_T$. When averaged over an input batch of size $N$, with $M = 2$ output classes (*fall* or *nonfall*), the $L_{nll}$ results in:

$$L_{nll} = \frac{1}{N} \sum_{\mu}^{N} \left[ - \sum_{i}^{M} y_i log(p_i) \right]_{\mu} , \tag{1}$$

where $y_i$ is the true probability. To constrain the membrane potential fluctuations around $U_i = 0$, a regularizing term $L_{reg}$ was formulated as follows:

$$L_{reg} = -\frac{1}{N} \sum_{\mu}^{N} \Big( \log\big(1 + \exp(U_\mu)\big) + \log\big(1 + \exp(-U_\mu)\big) \Big). \tag{2}$$

The total training loss is defined as $L_{tot} = L_{nll} + \alpha L_{reg}$, where $\alpha = 0.5$ was used for the surrogate gradient descent.

The RSNN hyperparameters were determined by means of a Hyper-Parameters Optimization (HPO) procedure implemented using the Neural Network Intelligence (NNI) toolkit [17]. A total of $N = 250$ experiments were performed over the ten train/validation splits to find the optimal subset of network hyperparameters with the Anneal tuner algorithm.

***Events + LSTM:*** An LSTM network was used to process the temporal data. Specifically, we implemented a three-layer LSTM model following the RSNN structure, with the first layer comprising LSTM nodes with an input shape of (1024, 700), Fig. 2, followed by two fully connected layers. The number of hidden nodes was set equal to the number of neurons in the RSNN hidden layer.

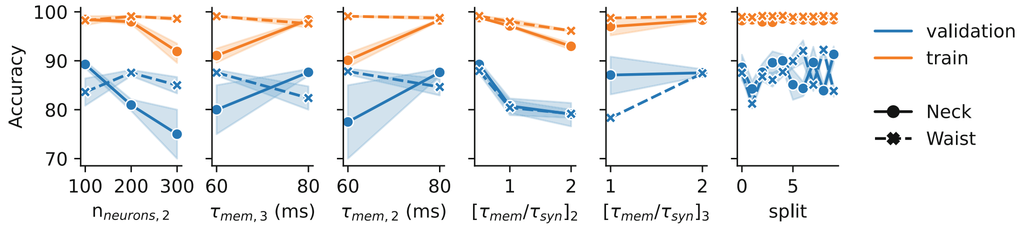

**Fig. 4.** Hyper-parameters Optimization. RSNN classification accuracy as a function of the model hyperparameters.

### B) Classifiers for the Frame-Based Data

***CNN + LSTM:*** We extracted 20 evenly-sampled frames from each clip within the time window $[T^* - \Delta_T/2, T^* + \Delta_T/2]$ chosen for the event-based models. This data was then fed to the pipeline comprising a ResNet50 CNN, followed by a downstream LSTM classification head.

## 4 Experimental Results

**Baseline Classifier.** We first evaluated the frame-to-event data conversion by feeding the "time-collapsed" event-based data to a linear SVM. The validation accuracy, measured over the 10 validation sets, is $81 \pm 4\%$ for the N-data and $85 \pm 5\%$ for the W-data. Given the time-based nature of the event-based data, which is inherently removed in the "time-collapsed" representation, we anticipate that our results are highly dependent on the choice of the specific time window $[T^* - \Delta, T^* + \Delta]$ used to crop the initial video. However, this preliminary analysis served only to confirm that the information content needed to address the fall-detection problem was still preserved after the frame-to-event conversion.

**Hyper-Parameter Optimization.** Figure 4 shows the results of the HPO experiments for the RSNN, with the classification accuracy reported as a function of the optimized hyperparameters, i.e., the membrane time constant of the hidden and output layers ($\tau_{mem,2}$, $\tau_{mem,3}$, respectively), the ratio between membrane time constant and synaptic time constant for hidden and output layers ($[\tau_{mem}/\tau_{syn}]_2$, $[\tau_{mem}/\tau_{syn}]_3$, respectively) and the number of neurons in the hidden layer ($\mathrm{n}_{\mathrm{neurons},2}$). Each data point is the result of one experiment running on one randomly selected train/validation split. Note that for both W- and N-data, the architectures converge to a time constant ratio $[\tau_{mem}/\tau_{syn}]_2 = 0.5$ and $[\tau_{mem}/\tau_{syn}]_2 = 2$, respectively.

**Comparative Analysis.** The results of our comparative analysis are shown in Table 1, which reports all the figures of merit taken into account.

*Performance:* The performance analysis on the classification accuracy shows that the CNN operating on the high-resolution RGB frames reaches the best accuracy

**Table 1.** Results of fall detection by different data types and methods. The number of operations are with respect to the inference stage, #Par indicates the number of trainable parameters, and $N_t$ refers to the number of time steps.

| Data | Methods | Operations | | # Par* | Accuracy† | |
|---|---|---|---|---|---|---|
| | | MACs ($N_t$) | ACs ($N_t$) | | Waist | Neck |
| events | LSTM | $5.7 \times 10^6 N_t$ | 0 | $0.5 \times 10^6$ | 0.779(±0.02) | 0.897(±0.02) |
| events | RSNN | $2.3 \times 10^3 N_t$ | $4 \times 10^5 N_t$ | $0.1 \times 10^6$ | 0.760(±0.04) | 0.810(±0.03) |
| frames | CNN+LSTM | $8.3 \times 10^{10} + 1.6 \times 10^5 N_t^{\diamond}$ | 0 | $2.5 \times 10^7$ | 0.997(±0.01) | 0.981(±0.01) |

$*$ With $\mathrm{n}_{\mathrm{neurons},2} = 100$ for LSTM and RSNN.

$\diamond$ MACs of the pretrained Resnet50 are accounted once as the model operates on all input frames at once.

† Reported with the set of hyper-parameters obtained with HPO.

score, with 99.7% accuracy for W-data and 98.1% for N-data, and outperforms both event-based methods. This, however, comes at the cost of a larger number of trainable parameters (i.e., 2 orders of magnitude larger). By contrast, the performance gap of the RSNN model, when compared to the LSTM operating on the same input resolution, is significantly lower (2% points for W-data and 9% points for N-data).

*Computational Cost:* We measured the number of trainable parameters, and the number of computations required per time step in terms of accumulation (AC) and MAC operations. Given the binary nature of spiking inputs, spikes transmission in a RSNN can be approximated as ACs operations, which are less power consuming than MACs [18]. The computation of MACs and ACs for RSNN and LSTM is based on the theoretical analysis presented in [19], while the operations count for Resnet50 is calculated using THOP, a tool for operation counting, validated by previous studies [20]. Compared to the LSTM, which operates on the same type of event-based input, the RSNN requires 2000× less MACs in the inference stage.

## 5 Discussion and Conclusion

We propose a novel high-energy efficient methodology for fall detection. We performed a comparative analysis of this event-based approach coupled with two types of classifiers, namely RSNN and LSTM, versus a traditional CNN approach that operates on full-resolution RGB frames.

Prior to the comparative analysis, we used a linear classifier to assess whether the information content needed to detect falls is still preserved after the video-to-event conversion. In spite of the "time-collapsed" representation, the linear classifier achieved high (validation) accuracy when trained on the event count collected over the time window $\varDelta_T$. While this served as a benchmark performance for using the event-based data to detect falls, we conjecture that the performance of such a linear classifier is highly dependent on the specific choice for the time window $\varDelta_T$. Moreover, such an approach would not be ideal for learning spatio-temporal patterns, such as in fall detection, as it operates on data gathered over time windows and not on real-time streams of incoming data [14]. This is in contrast to time-based models, such as RSNNs and LSTMs, which offer promising candidate solutions to move towards online fall detection systems.

Hence, we evaluated the effectiveness of combining the event-based data stream with time-based approaches. Compared to the CNN, both event-based pipelines exhibit a drop in effectiveness. However, it is noteworthy that the CNN operates on input RGB frames of size $224 \times 224$ pixels as opposed to the $128 \times 128$ spatial resolution of the event-based pipeline, which was further downscaled to $32 \times 32$ pixels. As a result, the CNN approach comes at the cost of $200\times$ more trainable parameters than the shallow RSNN and far more MACs per step. In comparison to the LSTM approach, the number of MACs per step of the RSNN is three orders of magnitude lower based on a comparable shallow topology. This is however at the cost of a drop in accuracy. In future work, we will do a systematic analysis on the trade-off with respect to input size vs. accuracy vs. computational complexity.

In general, our research shows potential for energy-efficient pipelines for fall detection by using spike-based algorithms. Yet, based on the current experiments, in comparison to standard deep learning methods, the presented solution comes at the cost of decreased accuracy. To address this gap, in future work, we will investigate the incorporation of multiple recurrent hidden layers, explore different learning mechanisms for spike-based algorithms, and utilize Winner-Takes-All layers for real-time network predictions. To fully leverage the computational efficiency of RSNNs, future studies may focus on deploying them on specialized neuromorphic hardware that takes advantage of their event-based nature. Additionally, before deploying on an embedded system, we aim to compare our simulated event-based data with a new dataset collected using an eDVS.

We anticipate that the envisioned end-to-end event-based pipeline for fall detection can deliver novel embedded solutions for fall detection, with promising performance vs. energy efficiency trade-offs and enhanced privacy and security by processing data directly on the edge.

## References

1. Elliott, S., Painter, J., Hudson, S.: Living alone and fall risk factors in community-dwelling middle age and older adults. J. Commun. Health **34**, 301–310 (2009)
2. Nooruddin, S., Islam, M.M., Sharna, F.A., Alhetari, H., Kabir, M.N.: Sensor-based fall detection systems: a review. J. Ambient Intell. Humaniz. Comput. 1–17 (2022)
3. World Health Organization. Ageing, & Life Course Unit: WHO global report on falls prevention in older age. World Health Organization (2008)
4. Igual, R., Medrano, C., Plaza, I.: Challenges, issues and trends in fall detection systems. Biomed. Eng. Online **12**(1), 66 (2013)
5. Gallego, G., et al.: Event-based vision: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **44**(1), 154–180 (2020)
6. Wang, X., Ellul, J., Azzopardi, G.: Elderly fall detection systems: a literature survey. Front. Robot. AI **7**, 71 (2020)

7. Ma, X., Wang, H., Xue, B., Zhou, M., Ji, B., Li, Y.: Depth-based human fall detection via shape features and improved extreme learning machine. IEEE J. Biomed. Health Inform. **18**(6), 1915–1922 (2014)
8. Huang, C.: Event-based timestamp image encoding network for human action recognition and anticipation. In: International Joint Conference on Neural Networks, pp. 1–9. IEEE (2021)
9. Moreno-Rodríguez, F.J., Traver, V.J., Barranco, F., Dimiccoli, M., Pla, F.: Visual event-based egocentric human action recognition. In: Pinho, A.J., Georgieva, P., Teixeira, L.F., Sánchez, J.A. (eds.) IbPRIA 2022. LNCS, vol. 13256, pp. 402–414. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-04881-4_32
10. Casares, M., Ozcan, K., Almagambetov, A., Velipasalar, S.: Automatic fall detection by a wearable embedded smart camera. In: International Conference on Distributed Smart Cameras, pp. 1–6 (2012)
11. Ozcan, K., Mahabalagiri, A.K., Casares, M., Velipasalar, S.: Automatic fall detection and activity classification by a wearable embedded smart camera. J. Emerg. Sel. Top. Circuits Syst. **3**(2), 125–136 (2013)
12. Hu, Y., Liu, S.C., Delbruck, T.: v2e: from video frames to realistic DVS events. In: Conference on Computer Vision and Pattern Recognition, pp. 1312–1321 (2021)
13. Conradt, J., Berner, R., Cook, M., Delbruck, T.: An embedded AER dynamic vision sensor for low-latency pole balancing. In: 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, pp. 780–785. IEEE (2009)
14. Muller-Cleve, S.F., et al.: Braille letter reading: a benchmark for spatio-temporal pattern recognition on neuromorphic hardware. Front. Neurosci. **16**, 951164 (2022)
15. Zenke, F., Vogels, T.P.: The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. Neural Comput. **33**(4), 899–925 (2021)
16. Bouanane, M.S., Cherifi, D., Chicca, E., Khacef, L.: Impact of spiking neurons leakages and network recurrences on event-based spatio-temporal pattern recognition. arXiv preprint arXiv:2211.07761 (2022)
17. Microsoft: Neural Network Intelligence (2021). https://github.com/microsoft/nni
18. Horowitz, M.: 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), pp. 10–14. IEEE (2014)
19. Yin, B., Corradi, F., Bohté, S.M.: Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. Nat. Mach. Intell. **3**(10), 905–913 (2021)
20. Hazarika, A., Poddar, S., Nasralla, M.M., Rahaman, H.: Area and energy efficient shift and accumulator unit for object detection in IoT applications. Alex. Eng. J. **61**(1), 795–809 (2022)