frontiers | Frontiers in Neuroscience

TYPE Original Research
PUBLISHED 11 May 2023
DOI 10.3389/fnins.2023.1160034

OPEN ACCESS

EDITED BY
Anup Das,
Drexel University, United States

REVIEWED BY
Vivek Parmar,
Indian Institute of Technology Delhi, India
Priyadarshini Panda,
Yale University, United States

*CORRESPONDENCE
Javier Cuadrado
✉ javier.cuadrado@cnrs.fr

RECEIVED 06 February 2023
ACCEPTED 13 April 2023
PUBLISHED 11 May 2023

CITATION
Cuadrado J, Rançon U, Cottereau BR, Barranco F and Masquelier T (2023) Optical flow estimation from event-based cameras and spiking neural networks.
Front. Neurosci. 17:1160034.
doi: 10.3389/fnins.2023.1160034

Optical flow estimation from event-based cameras and spiking neural networks

Javier Cuadrado[1]*, Ulysse Rançon[1], Benoit R. Cottereau[1,2], Francisco Barranco[3] and Timothée Masquelier[1]

[1]CerCo UMR 5549, CNRS – Université Toulouse III, Toulouse, France, [2]IPAL, CNRS IRL 2955, Singapore, Singapore, [3]Department of Computer Engineering, Automatics and Robotics, CITIC, University of Granada, Granada, Spain

Event-based cameras are raising interest within the computer vision community. These sensors operate with asynchronous pixels, emitting events, or "spikes", when the luminance change at a given pixel since the last event surpasses a certain threshold. Thanks to their inherent qualities, such as their low power consumption, low latency, and high dynamic range, they seem particularly tailored to applications with challenging temporal constraints and safety requirements. Event-based sensors are an excellent fit for Spiking Neural Networks (SNNs), since the coupling of an asynchronous sensor with neuromorphic hardware can yield real-time systems with minimal power requirements. In this work, we seek to develop one such system, using both event sensor data from the DSEC dataset and spiking neural networks to estimate optical flow for driving scenarios. We propose a U-Net-like SNN which, after supervised training, is able to make dense optical flow estimations. To do so, we encourage both minimal norm for the error vector and minimal angle between ground-truth and predicted flow, training our model with back-propagation using a surrogate gradient. In addition, the use of 3d convolutions allows us to capture the dynamic nature of the data by increasing the temporal receptive fields. Upsampling after each decoding stage ensures that each decoder's output contributes to the final estimation. Thanks to separable convolutions, we have been able to develop a light model (when compared to competitors) that can nonetheless yield reasonably accurate optical flow estimates.

KEYWORDS

optical flow, event vision, spiking neural networks, neuromorphic computing, edge AI

1. Introduction

Computer vision has become a domain of major interest, both in research and in industry. Indeed, thanks to the development of new technologies, such as autonomous vehicles or self-operating machines, algorithms able to perceive the environment have proven to be key to achieving the desired level of performance. Among the numerous visual features these algorithms can estimate, optical flow (the pattern of apparent motion on the image plane due to relative displacements between an observer and his environment) remains one of paramount importance. Indeed, this magnitude is directly linked with depth and egomotion, and its rich, highly temporal information is precious for advanced computer vision applications, e.g., for obstacle detection and avoidance in autonomous driving systems. Given the severe safety constraints associated with this kind of critical systems, accuracy, and reliability are key to achieving successful models. However, achieving high levels of performance is not enough: the increasing concern about energy consumption motivates us to seek the most efficient model possible, all while retaining high-performance standards.

In the search of an energy-efficient way to estimate optical flow, we decided to focus our interest on event cameras. Unlike their regular, frame-based counterpart, this kind of sensor is composed of independent pixel processors, each firing asynchronous events when the variation of the detected luminance since the previous event reaches a given threshold, being this event of positive polarity if the brightness has increased, and of negative polarity otherwise. This behavior translates into enormous energy savings: whereas conventional frame-based cameras are forced by design to output a frame at a fixed frequency, event cameras do not trigger any events for static visual scenes. Furthermore, they show a higher dynamic range, which allows them to avoid problems such as image artifacts (e.g., saturation after leaving a tunnel while driving), and a lower latency than regular cameras, which makes them particularly suitable for challenging, highly dynamic tasks (event sensors do not suffer from motion blur, unlike their frame-based counterparts). Nevertheless, they can also be less expressive: event cameras only provide information regarding changes in luminance, and not about the luminance itself. Furthermore, most event cameras discard color information, although some devices exist with independent firing for RGB formation at each pixel, like the Color-DAVIS346 event camera used by Scheerlinck et al. (2019) to generate their CED Dataset. Finally, event cameras usually have lower spatial resolution than regular cameras, although recent technological developments are bridging this gap (e.g., PROPHESEE, 2021).

In search of energy efficiency, the choice of the sensor is not enough: the optical flow prediction algorithm itself also has to be as efficient as possible to achieve our goal. That is why we have resorted to Spiking Neural Networks (SNNs) to develop our model. These bio-inspired algorithms, heavily inspired by the brain, consist of independent units (neurons), each of them with an inner membrane potential, which can be excited or inhibited by pre-synaptic connections. When their inner potential reaches a certain, predefined firing threshold, one spike is sent to the post-synaptic neurons, and the membrane potential is reset. Since energy consumption on dedicated hardware is linked to spike activity, which is usually much sparser than standard analog neural networks activations, SNNs represent a more energy-efficient alternative. Moreover, in the absence of movement, no input events would be produced and fed to the network, which in turn would not trigger any spikes, and a zero-optical flow prediction would be achieved (which is indeed the desired behavior, since no input events can only be achieved by a lack of relative motion).

Finally, optical flow being a highly temporal task, incorporating temporal context into our vision model is key to achieving acceptable levels of performance. Two alternatives exist: using stateful units within the network (e.g., LSTMs, GRUs or taking advantage of the intrinsic memory capabilities in the case of SNNs), or explicitly handling the temporal dependencies with convolutions over consecutive frames along a temporal axis. Exploiting spiking neuron inherent temporal dynamics has proven to be an extremely challenging task to achieve, and we have therefore opted for the second alternative.

To sum up, the main contributions of this article are:

- A novel angular loss, which can be used with standard MSE-like functions and which helps the network to learn an intrinsic spatial structure. To the best of our knowledge, we are the first to ever use such a function for optical flow estimation.
- 3d-encoding of input events over a temporal dimension, leading to increased optical flow estimation accuracy.
- A hardware-friendly downsampling technique in the form of maximum pooling, that further improves the model's accuracy.
- A spiking neural network which can be implemented on neuromorphic chips, therefore taking advantage of their energy efficiency.

2. Related work

Ever since their introduction, event cameras have been gaining ground within the computer vision community, and increasing efforts have been made to develop computer algorithms based on event data. As such, different datasets have emerged in order to solve different kinds of computer vision problems, like the DVS128 Gesture Dataset by Amir et al. (2017) for gesture classification, or the EVIMO Dataset by Burner et al. (2022) for motion segmentation and egomotion estimation. Despite this interest in event vision, the significant investment that event cameras represent for most research centers and companies has led to the development of event data simulators such as CARLA by Dosovitskiy et al. (2017), as well as algorithms to perform video-to-events conversion, like the model proposed in Gehrig et al. (2021b). While lacking the intrinsic noise event data usually presents, these artificial data can nonetheless be used to efficiently pre-train computer vision neural networks, e.g., Hidalgo-Carrió et al. (2020) pretraining their model for depth estimation on a synthetic set of event data.

Nonetheless, for real-world applications (e.g., gesture recognition, object detection, clustering, etc.), true event recordings are preferred because simulators are still lacking realistic event noise models. Concerning depth and/or optical flow regression, two datasets have currently established themselves as the go-to choices: the MVSEC Dataset by Zhu et al. (2018a), and the DSEC Dataset by Gehrig et al. (2021a). While all of these datasets have proven invaluable to develop event-based computer vision algorithms, there is still an enormous gap between event-based and image-based publicly available datasets, and many authors are still forced to develop their own. For example, Cordone et al. (2022) generated their own classification data from de Tournemire et al. (2020) to account for the additional "pedestrian" class.

Most models so far have either been standard Analog Neural Networks (ANNs) like Gehrig et al. (2021b), exploiting gated-recurrent units to achieve state-of the art accuracy on DSEC, or hybrid analog-spiking neural networks like Lee et al. (2022), combining a spiking encoder with an additional analog encoder for grayscale images, followed by a standard ANN. Other models have tried to leverage the temporal context by feeding the network with not only the events themselves, but also information on event timestamps, like the EVFlowNet model presented in Zhu et al. (2018b). More recently, Zhang et al. (2022) showed temporal information to be a key in accurately estimating both optical flow and depth, achieving top results in the MVSEC and the DSEC datasets thanks to their implementation of non-spiking leaky

integrators with learnable per-channel time constants. While all of these models do indeed achieve good levels of performance on their test sets, none of them manage to take advantage of the neuromorphic-friendly nature of event data, since analog blocks or additional non-spiking information prevent a deployment on neuromorphic chips.

More interesting to this work are spiking neural networks applied to event vision, be it for depth or for optical flow estimation. As far as optical flow is concerned, it is worth citing the works of Hagenaars et al. (2021), which achieves state-of-the-art levels of performance on the MVSEC Dataset with a fully spiking architecture. More recently, Kosta and Roy (2022) showed that spiking neural networks can indeed compete with their analog counterparts in terms of accuracy, showing top results both in the MVSEC and in the DSEC Dataset. Finally, Zhang et al. (2023) achieves a remarkable accuracy on the MVSEC Dataset with a U-Net-like architecture and a self-supervised learning rule. However, all of these models are not implementable on neuromorphic hardware, since they either use upsampling techniques which are incompatible with the spiking nature of these devices (e.g., bilinear upsampling), or re-inject intermediate, lower-scale analog optical flow predictions, thereby violating the spiking constraint by introducing floating point values in an otherwise binary model. In addition, the choice of a self-supervised learning rule, usually linked to a photometric loss function presented in Yu et al. (2016), means that optical flow estimations are only provided for pixels where events occurred, therefore creating non-dense flow maps. Looking at depth prediction though, we do find some interesting strategies for fully deployable neuromorphic models. Finally, authors in Rançon et al. (2022) presented in their StereoSpike model a fully-spiking, hardware-friendly network achieving remarkable accuracy on the MVSEC Dataset, thanks to stateless spiking neurons that have greatly inspired our work.

While we have focused on optical flow and depth predictions with event cameras, there have also been preceding works achieving top results on other computer vision tasks using event datasets and spiking neural networks. Such is the case of the works of Kim et al. (2022), who performed semantic segmentation via supervised training of a SNN, or the method described in Kirkland et al. (2022) that addressed instance segmentation on event data using a biologically-plausible learning strategy.

3. Materials and methods

3.1. Training dataset

Our study focuses on driving scenes, and we chose the DSEC Dataset by Gehrig et al. (2021a) to train our model. Unlike previous state-of-the-art datasets, such as the Muti-Vehicle Stereo Event Camera (MVSEC) Dataset by Zhu et al. (2018a), which provided different working scenarios (indoors/outdoors, day/night, and four possible vehicle configurations: pedestrian, motorbike, car and drone), the DSEC dataset only consists of driving scenario sequences. However, it provides higher-quality ground-truth labels, thanks to the finer processing of the LIDAR measurements. In addition, this dataset also includes masks for invalid pixels, i.e., pixels where the optical flow ground-truth is unknown. As such, our metrics have only been evaluated on the valid pixels. Furthermore, this dataset provides an open benchmark to submit the results, which we used to determine our test metrics and compare ourselves to other works.

3.2. Input event representation

Event cameras produce an asynchronous event e_i when the luminance variation at a given pixel reaches a given threshold:

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

where (x_i, y_i) are the coordinates of the pixel emitting the event, t_i the event's timestamp, and p_i its polarity (+1 if luminance increases, and −1 otherwise). However, in order to perform our training, we are forced to work with a discrete time model, so a pre-processing of this event stream has to be made. We therefore transform the input event stream into a sequence of frames of a given length in miliseconds, that we call "input histograms". These frames consist of a two-channel ($C = 2$) tensor of size (C, H, W), where H and W represent the camera's resolution, i.e., the number of input pixels and their position in the camera. At each pixel, the first channel represents the number of positive input events that have been triggered in that particular pixel during the frame's duration, and the second channel represents the number of negative events. A representation of a one-channel input frame can be found in Figure 1. While not a binary representation, like the representation paradigm presented in Cordone et al. (2021), our choice is more expressive, since event counts account for pixel relative importance and therefore provide richer spatio-temporal information.

We acknowledge that this frame-based approach increases the model's latency, since event sensors can virtually function in continuous time. However, it is imposed by the nature or our training, and is a widespread technique for event-based learning (see Gallego et al., 2022 on event representations). Moreover, we can leverage the latency reduction by our frame duration choice: the input stream being a continuous sequence of events, we are free to cumulate them in windows of the desired duration.

3.3. Spiking neuron model

For our network, we chose a simple neuron model that can be easily implemented with open-source Python libraries, in addition to being much less computationally expensive than closer-to-nature neuron mathematical models. This model is the McCulloch and Pitts (1943). It was implemented using the Spikingjelly library, developed and maintained by Fang et al. (2020), due to their full integration with the Pytorch library.

Our model is based on a stateless approach: the neuron's potential is reset after each forward pass. Indeed, the mathematical neuron model presented by McCulloch and Pitts consists of stateless neurons with Heaviside activation functions. This is equivalent to stateless integrate-and-fire neurons, i.e., stateless artificial neurons working as perfect integrators, but which are reset at every time step. We therefore do not exploit the intrinsic memory capabilities of spiking neurons, but rather perform a

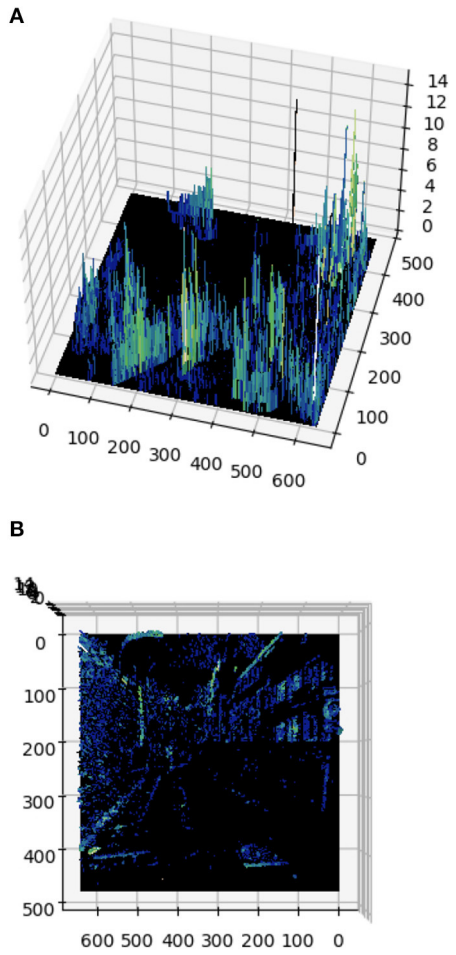

FIGURE 1
Example of an input frame for 1 polarity. **(A)** Event cumulation at each pixel for a given time interval. **(B)** Top view of the event frame. We can see that events are heavily linked to contours (e.g., a zebra crossing on the bottom part, or the windows on a building on the right side), while regions with constant luminance (e.g., the road or the sky) do not trigger events.

binary encoding of the information. While this approach may seem counter-intuitive, it actually further reduces energy consumption, since the reset operation is usually less energy demanding than the neuronal leak, and no resources have to be allocated to long-term memory handling. Consequently, we do not need to model such phenomenon, and as a result our neuron model is more hardware friendly than its leaky counterpart. Temporal context is handled by 3d convolutions in the encoder stages of the model, as we explain in the following section.

3.4. Network architecture

Our network is based on a U-Net-like architecture (Ronneberger et al., 2015). Indeed, U-Net has established itself as a reference model when full-scale image predictions are required, i.e., predictions at roughly the same resolution as the input data. Our architecture is shown in Figure 2. After a first convolution stage which increases the number of channels to 32 without modifying the input tensor size, each encoder stage halves the tensor width and height while doubling the number of channels. Conversely, each decoder stage doubles the tensor width and height, and halves the number of channels.

In order to increase the network expressivity, each decoder stage plays a role in the final prediction. Each decoder output is upsampled into a full-scale, two-channel tensor (x- and y-components of the optical flow estimation). All of the outputs equally contribute to the network's final estimation, which consists of the combination of successive coarse predictions. The loss function is evaluated after each update of the final neuron pool, thus forcing the network's prediction to be close to the ground-truth as early as the first coarse update. This approach has been introduced in Rançon et al. (2022) and proved to be beneficial to increasing the overall accuracy.

The main features of our network are the following:

- Inspired by Temporal-Convolutional Networks, presented in Lea et al. (2016) and Lea et al. (2017), we use three-dimensional convolutions for our data encoding. Consecutive input frames are combined by the temporal kernel via unpadded convolutions, decreasing the temporal dimension in size so it collapses to 1 when reaching the bottleneck. Acting as delay lines, they allow to explicitly handle the temporal dimension. The small temporal kernel size is able to capture short-term temporal relationships, while the increasing temporal receptive field due to consecutive convolutions along the temporal dimension accounts for long-term dependencies. Afterwards, the network architecture is fully two-dimensional. By default, the temporal kernel size we use is 5, which leads to a temporal receptive field of $21 \cdot 9ms = 189ms$ from the bottleneck and beyond.
- Skip connections between the encoder and the decoder consist of the last component of the temporal dimension at the corresponding encoding stage, since we believe the most recent event information to be the most relevant for optical flow estimation. We tested both sum and concatenate skip connections and found that concatenations led to the best estimations (these results are presented in Section 4.2).
- Given the relative importance of the residual blocks in the total number of parameters, and in search of the lightest possible model, we also analyzed the effect of reducing the number of residuals on the network's performance. We found that the best model only necessitated one residual, unlike other conventional U-Net-like architectures (e.g., Hagenaars et al., 2021).
- Downsampling in the encoding stages is performed via maximum pooling, instead of traditional strided convolutions, to account for spikes within the kernel's region, and not so much about individual spikes. This approach has proved to increase our model's performance. To the best of our knowledge, it is the first time this technique is used in a U-Net-like spiking neural network for dense regression. In addition, Gaurav et al. (2022) showed that this kind of downsampling strategy is supported by neuromorphic hardware.

10.3389/fnins.2023.1160034

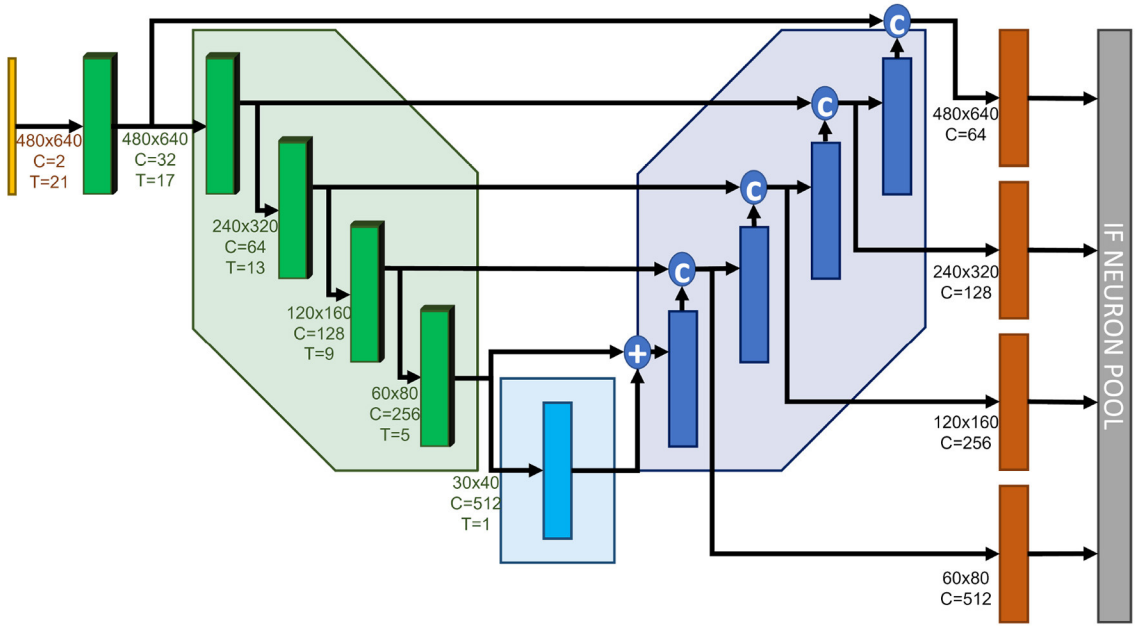

FIGURE 2
Our proposed network architecture. 3D Encoders ensure the incorporation of a temporal context within the model. Downsampling is performed via max. pooling to account for spatial spike activity. Each decoding stage is upsampled to contribute to the final network prediction.

- Since our final aim is to develop a model that could be implemented on a neuromorphic chip, the whole upsampling operation is performed via Nearest Neighbor upsampling, which preserves hardware friendliness. Indeed, while other widespread techniques, such as Bilinear Upsampling, interpolate each "pixel", Nearest Neighbor Upsampling simply copies each value into a tensor of an increased size, without modifying it. For further illustration, a graphic representation of both upsampling techniques can be found in the Supplementary material (Supplementary Figure 1).
- To further decrease the model's weight, we used depth- and point-wise separable convolutions (see e.g., Chollet, 2017) everywhere in the model. These convolutions do not only decrease the model's number of parameters, but also reduce the model's overfitting, therefore increasing its performance on unseen data.

It is important to specify that our approach is integer rather than binary-based, since some of our skip connections are additions instead of concatenations, and our bottleneck's architecture is based on tensor sums. Nevertheless, our approach remains hardware-friendly, because:

- If the processing were asynchronous and event-driven, then the spikes arriving through the residual connection would typically arrive before the others. Thus, if there were two spikes, one from the residual and one from the normal connection, instead of doing an explicit ADD, both spikes could be fed through the same synapse, and each spike would cause an increment of w (instead of adding the two spikes to get 2 and then multiplying by w to get the increment). Moreover, even if the spikes arrived synchronously, they would be processed sequentially using FIFO.
- Concatenation is equivalent to addition as a skip connection if the weights are duplicated and kept tight. Indeed, if there were two spikes, one from the residual and one from the "normal" connection, instead of doing $2 \cdot w$, the algorithm would perform $w + w$. Since the duplicated weights would be tight, the number of trainable parameters would be the same, and both operations would be equivalent.

3.5. Supervised learning method

Our model was trained with supervised learning using the surrogate gradient descent, using a sigmoid function as our surrogate gradient model. The ground-truth optical flow values were those provided in the DSEC database. While traditional self-supervised methods restrict their optical flow processing to pixels where events occurred (e.g., Zhu et al., 2018b, Hagenaars et al., 2021, or Kosta and Roy, 2022, to cite a few examples), our approach permits dense estimations (thanks to surrogate gradient learning). We trained our model on the valid pixels given by the dataset masks at each timestep.

Our loss function included two terms:

- A standard MSE-like loss between the value of the predicted flow and its corresponding ground truth, with the following formula:

$$L_{mod} = \frac{\sum^{N_{pixels}} \sqrt{(pred_x - gt_x)^2 + (pred_y - gt_y)^2}}{N_{pixels}} \quad (2)$$

The term N_{pixels} represents the number of valid pixels to be trained at each timestep.

- In addition to a penalization in modulus discrepancy between the vectors, we explicitly encourage the optical flow direction to be the same between the ground-truth and the prediction. This term has proven to be key to reduce noise in optical flow predictions, since pixels with low optical flow values consistently yield small modulus loss values regardless of their direction. We used the following formula:

$$L_{ang} = \frac{\sum^{N_{pixels}} acos(c\theta)}{N_{pixels}} \quad / \quad c\theta = \frac{\vec{gt} \cdot \vec{pred} + \epsilon}{|\vec{gt}| \cdot |\vec{pred}| + \epsilon} \tag{3}$$

where $c\theta$ is the cosine of the error angle between the predicted and the ground-truth flow, and epsilon is a small parameter ($\epsilon = 10^{-7}$) to ensure that no errors are found within the code during execution. Furthermore, the values of $c\theta$ are clamped between $(-1 + \epsilon, 1 - \epsilon)$ for the same reason.

The final loss function used to train the model is:

$$L = \lambda_{mod} \cdot L_{mod} + \lambda_{ang} \cdot L_{ang} \tag{4}$$

From preliminary tests, we found that $\lambda_{mod} = \lambda_{ang} = 1$ yields good results, and we therefore decided to use these values.

As explained in Section 3.4 (Network Architecture), each decoder's output plays a role in the final optical flow estimation. As such, and in order to encourage accuracy since the first decoder's upsampling, the loss function is evaluated for each consecutive contribution to the final pool. After each upsampling of the decoder's output, the inner potentials of an IF layer are updated, and the loss is evaluated on those potentials equivalent to summing the spikes out of each decoder stage weighted by the corresponding intermediary prediction layer.

Finally, in order to perform the back-propagation in our supervised training method, we resorted to surrogate gradient learning, introduced in Neftci et al. (2019), and already implemented in the SpikingJelly library (Fang et al., 2020).

3.6. Training details

All of our calculations were performed on either NVIDIA A40 GPUs, or in Tesla V100-SXM2-16GB GPUs belonging to the French regional public supercomputer CALMIP, owned by the Occitanie region.

Trainings were realized with a batch size of 1, since it is the optimal value we have found for our task. Although unconventional, this result is in line with the one found in Rançon et al. (2022), where a batch size of one was found optimal for depth regression from event data using stateless spiking neurons. We used an exponential learning rate scheduler, and have implemented random horizontal flip as a data augmentation technique to improve performance. Furthermore, thanks to our stateless approach, we were able to train our network with shuffled samples, instead of being forced to use the input frames sequentially.

4. Results

We divided our dataset into a train and a validation split, and our performance levels are reported with regard to the validation set. The exact sequences used in each split can be found in the Supplementary material. Nevertheless, we resort to the official DSEC benchmark to compare ourselves to the state-of-the-art, since it represents an objective, third-party test set. We now proceed to present the results we obtained in our studies. Due to the number of tests that we have run, all of the corresponding plots are provided in the Supplementary material.

4.1. Finding the optimal kernel size

Convolutional neural networks have regained the interest of the deep learning community during the past few years, thanks to their ability to capture spatial relations within their kernel. Recently, increased kernel sizes have been replacing the traditional 3×3 formula, with examples as relevant as Liu et al. (2022), which uses 7×7 kernels. Ding et al. (2022) presents a method to scale up the kernel size to 31×31, and Liu et al. (2023) goes even further and proposes to go up to 51×51 for the spatial kernel size, although both of these methods rely on sparsity and re-parametrization to achieve their goal. Starting from a naive U-Net like model, we started our research by trying to optimize our spatial kernel size. In the end, our results do match those presented in Ding et al. (2022), showing that 7×7 kernels are optimal. Indeed, further increasing the kernel size makes computational time explode, while accuracy plateaus. We therefore decided to adopt a 7×7 kernel in the spatial dimension for our model.

Next, we optimized the temporal kernel size, directly linked with the number of frames that we input to our model. Since we want the temporal dimension to collapse to one in the bottleneck thanks to unstrided convolutions in the temporal dimension, a larger kernel size naturally requires a greater number of frames, and therefore a heavier model. Nonetheless, it also takes into account a longer temporal context, which may be beneficial for the network's accuracy. As such, we tested our simple model for temporal kernel sizes of 3 (11 input frames), 5 (21 input frames), and 7 (31 input frames). Our results show that increasing the kernel size up to 5 does indeed boost the model's accuracy, but going beyond this size does not translate into an accuracy improvement. Thus, a temporal kernel size of 5 was chosen for the 3d convolutions in our model.

4.2. Finding the best network architecture

In order to find the best network architecture, we evaluated two possible options:

- We compared sum vs. concatenate skip connections, since concatenate skip connections are easier to implement in neuromorphic hardware, but slightly increase the number of parameters in the network.

TABLE 1 Performance comparison for the different proposed architectures. All of the models have been trained for 35 epochs, using 21 input frames of 9 ms each.

Model	Mod loss	Angular loss	Num. params (M)
1 residual + sum skip connections	1.18	0.101	1.1
1 residual + cat skip connections	**1.10**	**0.094**	**1.2**
2 residual + sum skip connections	1.15	0.097	1.7
2 residual + cat skip connections	1.16	0.109	1.8

The bold values indicate the best architecture, its performances obtained for each of the metrics, and its number of parameters (in millions).

- Seeking to develop a model as light as possible, we also characterized the effect of the number of residuals in the network's bottleneck on the model's performance.

After training each of the models for 35 epochs, we found the best model to be the 1-residual network with concatenate skip connections, which amounts to a total of 1.22 million of parameters and leads to an accuracy of 1.1 pixels/second of average end-point error on our validation dataset, using 9 ms frames as an input in all cases. The results regarding the architecture optimization have been summarized in Table 1.

4.3. Optimizing the frame duration

Next, we focused our attention on the optimal frame duration to accurately estimate optical flow, i.e., the total temporal context the network processes when making a prediction. This parameter is directly linked with the latency the model can achieve, since optical flow estimations are only produced at the end of each frame (provided that the input tensors are treated as a sliding window, where only the last N=21 frames are considered).

We trained the network with frames of 4.5, 9, and 18 ms, respectively. Our results show that the optimal frame duration was 9 ms, followed by 18 ms, and finally we get the worst performance for frames of 4.5 ms. While it may seem counter-intuitive as a results, since 4.5 ms frames contain a finer representation of the event sequence, we believe this phenomenon is caused by the lack of overall temporal context. Indeed, by using short frames, the network is unable to extract longer-term dependencies, and therefore to accurately predict optical flow. That is also why we believe that 18m s frames, while coarser, do manage to better capture these long term dependencies, and therefor provide a more accurate estimation. These results, as well as all of the successive optimization studies we have performed, can be found on Table 2.

4.4. Comparison with the state-of-the-art

We trained our best architecture on the whole DSEC dataset for a total of 100 epochs. We evaluated our model on the official test set provided by DSEC. Results are shown on Table 3. In order to provide a fair comparison, we only included results on the official benchmark, and not those reported on custom validation sets. While still far from the best models, we demonstrate the power of spiking neural networks when applied to dense regression in computer vision, achieving good levels of performance with a fraction of parameters when compared to other models.

We also provide some of our model's results on the validation set, which can be found in Figure 3. These pictures show that, even if the network was not explicitly trained to distinguish image contours (since it was only trained on a selection of valid pixels at each timestep), it is nonetheless capable of extracting structural information within the scene and generalizing it, as illustrated in the rightmost images (unmasked predictions) for the given examples. These results demonstrate the model's general comprehension of the visual scene, and we believe represent a solid understanding of the pattern of motion.

TABLE 2 Performance comparison. The two best models have been tested for different slight modifications of the architecture, keeping the number of parameters mostly unchanged.

Model	Modifications	Mod loss	Angular loss
1 res + cat	-	**1.10**	**0.094**
2 res + sum	-	1.15	0.097
1 res + cat	4.5 ms frames	1.41	0.129
2 res + sum	4.5 ms frames	1.42	0.130
1 res + cat	18 ms frames	1.19	0.087
2 res + sum	18 ms frames	1.32	0.102
1 res + cat	Combined polarities	1.14	0.092
2 res + sum	Combined polarities	1.31	0.112

The bold values indicate the best architecture and its performances for each of the metrics. The underline value is the best performance on the AAE metric, obtained with an architecture that nonetheless did not manage to beat the top-performing model.

TABLE 3 Comparison with the state-of-the art, obtained from https://dsec.ifi.uzh.ch/uzh/dsec-flow-optical-flow-benchmark/.

Model	AEE (px/s)	AAE (deg)	Num. params (M)
E-RAF (Gehrig et al., 2021b)	**0.79**	**2.9**	5.3
Ours	1.71	6.3	**1.2**
MultiCM (Shiba et al., 2022)	3.47	14.0	-

The bold values indicate the best performance for each of the metrics, as well as the model with the lowest number of parameters.

4.5. Ablation studies

Several ablation studies have been performed on our best model to further demonstrate our claims, and we have gathered our conclusions in the following paragraphs. Plots containing all of these results can be found in the Supplementary material.

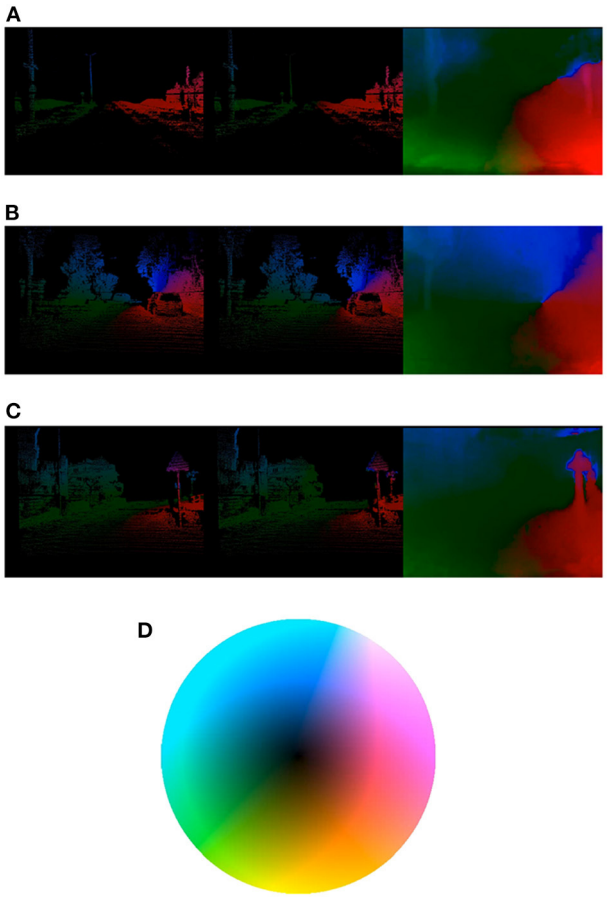

FIGURE 3
Example predictions of our best architecture on our validation set. For every picture, the leftmost image is the ground-truth, the middle image shows our masked estimation (only on valid pixels), and the rightmost image represents the unmasked estimation. **(A)** Optical flow discontinuities due to vertical artifacts within the visual scene. **(B)** The silhouette of the leftmost tree can be perceived on the unmasked optical flow map. **(C)** Traffic signs clearly distinguishable on the right side. **(D)** Depicts the chosen colormap for optical flow representation: Optical flow is encoded as an Lab image, where the luminance channel represents the absolute magnitude of the flow, and the a and b channels the different directions.

4.5.1. Pooling vs. convolutional downsampling

Our results show that using maximum pooling instead of strided convolutions is an efficient technique to downsample spiking data. We believe that the reason behind this behavior is that pooling is a way of densifying the tensors without changing their spiking nature.

4.5.2. 3d vs. 2d encoding

We also compared our baseline 3d model with an equivalent 2d model, where the 21 input frames have been fed to the network concatenated along the channel dimension, so that both models have the same temporal context. We found out that fully 2-dimensional models lead to decreased performance. We believe this is due to the fact that, by using 2d convolutions, all the temporal information is directly mixed during the first convolution stage, therefore hindering the network from finding long-term dependencies.

4.5.3. Loss function

We also analyzed the influence of the loss function on the final results obtained. We compared our proposed loss model to two single-term losses:

- One model with only the norm of the error vector, but without the angular loss term.
- One loss function with only a relative loss term:

$$L_{relative} = \frac{1}{N_{pixels}} \frac{\sum^{N_{pixels}} \sqrt{(pred_x - gt_x)^2 + (pred_y - gt_y)^2}}{\sqrt{gt_x^2 + gt_y^2} + \epsilon} \tag{5}$$

This model penalizes deviations in the prediction relative to the ground truth's norm, and should therefore be able to implicitly impose a restriction on angular accuracy.

Our results show that naively limiting the error's norm is not enough to achieve competitive results, and neither is limiting the relative error. Indeed, by introducing a more aggressive term in the loss function, we managed to force the network into implicitly learning the optical flow's structure, and therefore achieve better accuracy.

It is surprising that the network with the two losses reaches a lower L_{mod} than the network with L_{mod} only. This shows that the second network gets trapped in a local minima and that adding the L_{ang} loss helps to get out of it.

4.5.4. Effect of combining polarities on performance

Our next study on input representation has consisted in combining polarities into a single channel before feeding them to the model. Polarities being closely linked to phenomena like color or texture, we wish to study their influence on the final performance levels. Indeed, if we imagine a gray background with a black shape and a white shape following the same track, we would obtain opposite polarity fronts, while the optical flow pattern would be the same. We have therefore analyzed if polarities can be simply combined into a total per-pixel event count.

However, our results show that keeping separate channels for each polarities is beneficial for the network's performance. We believe this result is linked to the different dynamics linked to each of the polarities, since different thresholds lead to different behavior for luminance increments or decrements.

4.5.5. Skip connections in the bottleneck

Our final ablation study targeted the very first skip connection, i.e., connecting the last encoder with the first decoder. Having always kept it as a sum (slightly redundant, given the residual block architecture) because of the high number of channels, we have also tested transforming it into a cat skip connection. However, we found out that it decreases the network's performance while also

TABLE 4 Performance comparison on the MVSEC dataset (indoor sequences), showing per-sequence and total average end-point error in pixels per second.

Model	indoor_flying1	indoor_flying2	indoor_flying3	AEE sum
EV-FlowNet (Zhu et al., 2018b)	1.03	1.72	1.53	4.28
Zhu et al. (2019)	0.58	1.02	0.87	2.47
Spike-FlowNet (Lee et al., 2020)	0.84	1.28	1.11	3.23
Back to Event Basics$_{Evf}$ (Paredes-Vallés and de Croon, 2021)	0.79	1.40	1.18	3.37
Back to Event Basics$_{Fire}$ (Paredes-Vallés and de Croon, 2021)	0.97	1.67	1.43	4.07
XLIF-EV-FlowNet (Hagenaars et al., 2021)	0.73	1.45	1.17	3.35
XLIF-FireNet (Hagenaars et al., 2021)	0.98	1.82	1.54	4.34
Orchard et al. (2021)	0.83	1.22	0.97	3.02
Fusion-FlowNet (Lee et al., 2022)	<u>0.56</u>	0.95	0.76	2.27
Adaptive-SpikeNet (best ANN) (Kosta and Roy, 2022)	0.84	1.59	1.36	3.79
Adaptive-SpikeNet (best SNN) (Kosta and Roy, 2022)	0.79	1.37	1.11	3.27
FSFN$_{FP}$ (Apolinario et al., 2022)	0.82	1.21	1.07	3.10
FSFN$_{HP-ADC}$ (Apolinario et al., 2022)	0.85	1.29	1.13	3.27
Shiba et al. (2022)	**0.42**	**0.60**	**0.50**	**1.52**
Ours	0.58	<u>0.72</u>	<u>0.67</u>	<u>1.97</u>

Best result in bold, runner-up underlined. Starting from a model pre-trained on DSEC, we show state-of-the-art performance without modifying our pipeline.

increasing the number of parameters. We therefore decided to keep it as a sum for all of the architectures.

4.6. Model evaluation on the MVSEC dataset

In order to analyze the generalization capabilities of our method, we also tested our model on the Multi-Vehicle Stereo Event Camera Dataset (MVSEC), introduced in Zhu et al. (2018a). We started by analyzing our model performance on the indoor flying sequences. To do so, we took a model pre-trained for DSEC, and optimized its weights on the MVSEC Dataset over 35 epochs. We followed a training approach akin to the one adopted for the DSEC dataset, i.e., we only considered pixels with either zero-valued ground-truth (x- and y- components of the optical flow vector below a small threshold $thr = 1e-5$) or with unknown flow values as invalid, and only trained on valid pixels. The results we obtained, as well as a comparison with other state-of-the-art models, can be found in Table 4. We can see that we achieve state-of-the-art performance levels on these sequences when compared to other existing spiking neural networks, and top accuracy overall, even if our architecture has not been optimized for such a vehicle/scenario configuration.

Next, we also tested our model on the outdoor sequences on MVSEC: training on outdoor_day2, and evaluation on outdoor_day1. We present these results in Table 5. Although our model leads to competitive results on all of the MVSEC indoor sequences, it struggles to achieve competitive results on MVSEC outdoor sequences, both when starting from a pre-trained checkpoint or from scratch. We believe that this phenomenon is due to a combination of factors:

- Our network architecture, and most precisely the spatial kernel size, has been optimized for an optical flow prediction of 480 × 640 pixels. Nevertheless, the MVSEC dataset was recorded with a different event camera, and therefore may demand a different kernel size to achieve top performance levels.
- Our frame duration and overall temporal context have been designed for a specific camera configuration and resolution. Again, the use of a lower resolution camera leads to different optical flow dynamics, and therefore to potentially different temporal representation.
- Our training procedure (learning rate, scheduler, etc.) has not been designed for such a low-resolution estimation, and therefore further optimizations are needed to increase accuracy.
- Finally, the outdoor_day2 sequence of the MVSEC dataset, used for training on driving scenarios, consists of only 9 min of recording where high frequency vibrations are constantly affecting the event camera (see Zhu et al., 2018b). In addition, the event histograms are greatly impacted by events caused by reflections on the car dashboard. These noisy events may prevent from achieving competitive results

10.3389/fnins.2023.1160034

TABLE 5 Performance comparison on the MVSEC dataset (outdoor sequences), showing average end-point error in pixels per second.

Model	outdoor_day1 (px/s)
EV-FlowNet (Zhu et al., 2018b)	0.49
Zhu et al. (2019)	<u>0.32</u>
ECN$_{masked}$ (Ye et al., 2020)	**0.30**
Spike-FlowNet (Lee et al., 2020)	0.49
Back to Event Basics$_{Evf}$ (Paredes-Vallés and de Croon, 2021)	0.92
Back to Event Basics$_{Fire}$ (Paredes-Vallés and de Croon, 2021)	1.06
XLIF-EV-FlowNet (Hagenaars et al., 2021)	0.45
XLIF-FireNet (Hagenaars et al., 2021)	0.54
Fusion-FlowNet (Lee et al., 2022)	0.59
Adaptive-SpikeNet (best ANN) (Kosta and Roy, 2022)	0.48
Adaptive-SpikeNet (best SNN) (Kosta and Roy, 2022)	0.44
FSFN$_{FP}$ (Apolinario et al., 2022)	0.51
FSFN$_{HP-ADC}$ (Apolinario et al., 2022)	0.48
Shiba et al. (2022)	**0.30**
Ours	0.85

Best result in bold, runner-up underlined. While far from the top performing contributions, our base pipeline is able to learn to estimate optical flow from scratch, without any optimization to make it tailored to the dataset and camera.

in these sequences, since they are nonetheless responsible of inputting information to the network. In fact, only by masking that section in both the input event histogram and the associated ground-truth have we achieved training on this scenario: otherwise, the network oscillates without consistently increasing accuracy.

Nevertheless, our model achieves a certain level of learning on this condition, and we are convinced that better results could be obtained by optimizing the training pipeline for this scenario (specially the frame duration and the kernel sizes). Taking into account this learning, in conjunction with our competitive results on indoor flying scenarios, we believe that these results demonstrate the generalization capabilities of our approach, as well as its applicability in a variety of conditions.

5. Discussion

Briefly, we have presented a hardware-friendly, lightweight spiking model able to accurately estimate optical flow from event-based data collected by neuromorphic vision sensors. We propose an efficient temporal coding in the form of 3d convolutions in the encoder that increases the temporal receptive field of the deepest stages of the network. We also introduce a novel angular loss function that, in conjunction with a standard MSE-like loss, manages to boost performance by forcing the algorithm to learn the implicit spatial structure. We use maximum pooling as our downsampling strategy, thus densifying the tensors in a neuromorphic-friendly fashion. Moreover, the successive contributions of decoder outputs to the final prediction increase the network's expressivity, and allow us to achieve competitive results without resorting to intermediate prediction re-injections. Consequently, our model can be implemented in neuromorphic hardware, thus resulting in an extremely energy efficient model that can still achieve accurate predictions.

We believe our results contribute to promote spiking neural networks as energy-efficient, real-world alternatives to traditional computer vision systems, based on frame-based video treatment and/or complex sensor data. However, we acknowledge that work has yet to be done, since a lot of the intrinsic potential of SNNs, namely their inherent memory handling capabilities, has not been fully exploited in this study. Moreover, the convergence of our experiments to an optimal batch size of 1, while having indeed improved our model's performance, greatly hinders the training speed, since strategies such as data parallelization cannot be employed. We therefore believe that these results can be further improved, e.g., using techniques such as weight averaging or network pre-training.

Future research lines should focus on further combining different techniques in order to boost performance even further. For instance, exploiting the intrinsic memory of spiking neurons is indeed a potentially useful approach, but the increased computational power linked to unrolling a stateful computational graph makes the task challenging. Moreover, sensor fusion can also be explored as an alternative to boost performance, especially since most event cameras often also provide black and white images. This approach could increase the network's latency, as well as making neuromorphic implementation challenging. Furthermore, while temporal dependencies have been imposed a priori in our model, they could also be natively learnt by the network. The works of Khalfaoui-Hassani et al. (2021) present a way of increasing kernel sizes without an increment in network parameters, capable of achieving state-of-the-art performances. While only applied so far for 1- and 2-dimensional convolutions, their method could easily be adapted to our 3d approach.

Moreover, publicly available datasets usually lack challenging conditions, such as crossing pedestrians or vehicles, which can limit the network's generalization capabilities. While we believe that our proposed model is capable of understanding such situations (see Figure 3C, where traffic signals are easily recognizable), it would be desirable to train on more challenging scenarios.

Finally, we would like to address hardware efficiency and implementation. We acknowledge that our approach does not provide energy savings during training, since it is performed on GPUs using standard ANN learning techniques, and therefore suffers from the same energy consumption constraints as these networks (plus the added memory usage due to the stockage of the neuron's membrane potential. However, energy savings can be achieved when deployed on dedicated hardware, since they are more energy efficient than GPUs thanks to their spiking nature. Nonetheless, even if our model is hardware-friendly, and therefore theoretically implementable on dedicated hardware, more efforts can be dedicated toward making it easier to implement. Indeed, hardware mapping would benefit from weight quantization (which would require less bits to store each synaptic weight) or sparsity encouragement to fully exploit the neuromorphic

hardware advantages over GPUs. These techniques, presented by Orchard et al. (2021) and Apolinario et al. (2022), would not only reduce energy consumption, but also facilitate potential future implementations, and should be taken into account for actual on-chip deployment.

Data availability statement

Original datasets are available in a publicly accessible repository: https://dsec.ifi.uzh.ch/ (DSEC Dataset) and https://daniilidis-group.github.io/mvsec/ (MVSEC Dataset). The original contributions presented in the study are publicly available. This data can be found here: https://github.com/J-Cuadrado/OF_EV_SNN.

Author contributions

JC designed, programmed, ran the simulations, and wrote the main core of the article. All authors conceptualized the study and analyzed the results and provided comments to achieve the final version of the paper.

Funding

This research was supported in part by the Agence Nationale de la Recherche under Grant ANR-20-CE23-0004-04 DeepSee, by the Spanish National Grant PID2019-109434RA-I00/ SRA (State Research Agency /10.13039/501100011033), by a FLAG-ERA funding (Joint Transnational Call 2019, project DOMINO), and by the Program DesCartes and by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) Program.

Acknowledgments

This work was granted access to the HPC resources of CALMIP supercomputing center under the allocation 2022-p22020. The authors would also express their gratitude to the CerCo's NeuroAI team and specially to Mr. Khalfaoui-Hassani, for their constant support and insightful feedback. We would also like to thank Mr. Fang, developer of the SpikingJelly library, for the constant support he has provided during the whole timespan of this study.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnins.2023.1160034/full#supplementary-material

References

Amir, A., Taba, B., Berg, D., Melano, T., Mckinstry, J., Di Nolfo, C., et al. (2017). "A low power, fully event-based gesture recognition system," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE), 7388–7397. doi: 10.1109/CVPR.2017.781

Apolinario, M. P. E., Kosta, A. K., Saxena, U., and Roy, K. (2022). Hardware/software co-design with adc-less in-memory computing hardware for spiking neural networks. *arXiv preprint arXiv:2211.02167*. doi: 10.48550/arXiv.2211.02167

Burner, L., Mitrokhin, A., Fermüller, C., and Aloimonos, Y. (2022). Evimo2: an event camera dataset for motion segmentation, optical flow, structure from motion, and visual inertial odometry in indoor scenes with monocular or stereo algorithms. *arXiv preprint arXiv:2205.03467*. doi: 10.48550/arXiv.2205.03467

Chollet, F. (2017). "Xception: Deep learning with depthwise separable convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE), 1800–1807. doi: 10.1109/CVPR.2017.195

Cordone, L., Miramond, B., and Ferrante, S. (2021). "Learning from event cameras with sparse spiking convolutional neural networks," in *2021 International Joint Conference on Neural Networks (IJCNN)* (Shenzhen), 1–8. doi: 10.1109/IJCNN52387.2021.9533514

Cordone, L., Miramond, B., and Thierion, P. (2022). "Object detection with spiking neural networks on automotive event data," in *2022 International Joint Conference on Neural Networks (IJCNN)* (Padua), 1–8. doi: 10.1109/IJCNN55064.2022.9892618

de Tournemire, P., Nitti, D., Perot, E., Migliore, D., and Sironi, A. (2020). A large scale event-based detection dataset for automotive. *arXiv preprint arXiv: 2001.08499*. doi: 10.48550/arXiv.2001.08499

Ding, X., Zhang, X., Han, J., and Ding, G. (2022). "Scaling up your kernels to 31×31: Revisiting large kernel design in CNNs," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (New Orleans, LA: IEEE), 11953–11965. doi: 10.1109/CVPR52688.2022.01166

Dosovitskly, A., Ros, G., Codeville, F., Lopez, A., and Koltun, V. (2017). "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, eds S. Levine, V. Vanhoucke, and K. Goldberg (Mountain View, CA: PMLR), 1–16. Available online at: http://proceedings.mlr.press/v78/dosovitskiy17a/dosovitskiy17a.pdf

Fang, W., Chen, Y., Ding, J., Chen, D., Yu, Z., Zhou, H., et al. (2020). *Spikingjelly*. Available online at: https://github.com/fangwei123456/spikingjelly (accessed January 11, 2023).

Gallego, G., Delbruck, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., et al. (2022). Event-based vision: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* 44, 154–180. doi: 10.1109/TPAMI.2020.3008413

Gaurav, R., Tripp, B., and Narayan, A. (2022). Spiking approximations of the maxpooling operation in deep SNNs. *arXiv preprint arXiv:2205.07076*. doi: 10.48550/arXiv.2205.07076

Gehrig, M., Aarents, W., Gehrig, D., and Scaramuzza, D. (2021a). DSEC: a stereo event camera dataset for driving scenarios. *arXiv preprint arXiv:2103.06011*. doi: 10.1109/LRA.2021.3068942

Gehrig, M., Millhäusler, M., Gehrig, D., and Scaramuzza, D. (2021b). "E-RAFT: Dense optical flow from event cameras," in *2021 International Conference on 3D Vision (3DV)*. p. 197–206. doi: 10.1109/3DV53792.2021.00030

Hagenaars, J., Paredes-Valles, F., and de Croon, G. (2021). "Self-supervised learning of event-based optical flow with spiking neural networks," in *Advances in Neural Information Processing Systems, Vol. 34*, eds M. Ranzato, A. Beygelzimer, P. S. Liang, and J. W. Vaughan (Curran Associates), 7167–7169. Available online at: https://proceedings.neurips.cc/paper_files/paper/2021/file/39d4b545fb02556829aab1db805021c3-Paper.pdf

Hidalgo-Carrió, J., Gehrig, D., and Scaramuzza, D. (2020). "Learning monocular dense depth from events," in *2020 International Conference on 3D Vision (3DV)* (Fukuoka), 534–542. doi: 10.1109/3DV50981.2020.00063

Khalfaoui-Hassani, I., Pellegrini, T., and Masquelier, T. (2021). Dilated convolution with learnable spacings. *arXiv preprint arXiv:2112.03740*. doi: 10.48550/arXiv.2112.03740

Kim, Y., Chough, J., and Panda, P. (2022). Beyond classification: directly training spiking neural networks for semantic segmentation. *Neuromorph. Comput. Eng.* 2, 044015. doi: 10.1088/2634-4386/ac9b86

Kirkland, P., Manna, D., Vicente, A., and Di Caterina, G. (2022). Unsupervised spiking instance segmentation on event data using STDP features. *IEEE Trans. Comput.* 71, 2728–2739. doi: 10.1109/TC.2022.3191968

Kosta, A. K., and Roy, K. (2022). Adaptive-spikenet: event-based optical flow estimation using spiking neural networks with learnable neuronal dynamics. *arXiv preprint arXiv:2209.11741*. doi: 10.48550/arXiv.2209.11741

Lea, C., Flynn, M. D., Vidal, R., Reiter, A., and Hager, G. D. (2017). "Temporal convolutional networks for action segmentation and detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE), 1003–1012. doi: 10.1109/CVPR.2017.113

Lea, C., Vidal, R., Reiter, A., and Hager, G. D. (2016). "Temporal convolutional networks: A unified approach to action segmentation," in *Computer Vision – ECCV 2016 Workshops*, eds G. Hua and H. Jegou (Cham: Springer), 47–54. doi: 10.1007/978-3-319-49409-8_7

Lee, C., Kosta, A. K., and Roy, K. (2022). "Fusion-FLOWNET: Energy-efficient optical flow estimation using sensor fusion and deep fused spiking-analog network architectures," in *2022 International Conference on Robotics and Automation (ICRA)*. p. 6504–6510. doi: 10.1109/ICRA46639.2022.9811821

Lee, C., Kosta, A. K., Zhu, A. Z., Chaney, K., Daniilidis, K., and Roy, K. (2020). "Spike-flownet: event-based optical flow estimation with energy-efficient hybrid neural networks," in *Computer Vision–ECCV 2020: 16th European Conference* (Glasgow: Springer), 366–382.

Liu, S., Chen, Y., Chen, X., Chen, X., Xiao, Q., Wu, B., et al. (2023). More ConvNets in the 2020s: Scaling up Kernels Beyond 51x51 using Sparsity. *arXiv [Preprint]*. arXiv: 2207.03620. doi: 10.48550/arXiv.2207.03620

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. (2022). "A convNet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11976–11986. doi: 10.1109/CVPR52688.2022.01167

McCulloch, W. S., and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* 5, 115–133. doi: 10.1007/BF02478259

Neftci, E. O., Mostafa, H., and Zenke, F. (2019). Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Process. Mag.* 36, 51–63. doi: 10.1109/MSP.2019.2931595

Orchard, G., Frady, E. P., Rubin, D. B. D., Sanborn, S., Shrestha, S. B., Sommer, F. T., et al. (2021). "Efficient neuromorphic signal processing with loihi 2," in *2021 IEEE Workshop on Signal Processing Systems (SiPS)* (IEEE), 254–259. doi: 10.1109/SiPS52927.2021.00053

Paredes-Vallés, F., and de Croon, G. C. H. E. (2021). "Back to event basics: Self-supervised learning of image reconstruction for event cameras via photometric constancy," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE), 3445–3454. doi: 10.1109/CVPR46437.2021.00345

PROPHESEE (2021). *Metavision? Packaged Sensor*. Available online at: https://www.prophesee.ai/event-based-sensor-packaged/

Rançon, U., Cuadrado-Anibarro, J., Cottereau, B. R., and Masquelier, T. (2022). Stereospike: depth learning with a spiking neural network. *IEEE Access* 10, 127428–127439. doi: 10.1109/ACCESS.2022.3226484

Ronneberger, O., Fischer, P., and Brox, T. (2015). "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, eds N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi (Cham: Springer), 234–241. doi: 10.1007/978-3-319-24574-4_28

Scheerlinck, C., Rebecq, H., Stoffregen, T., Barnes, N., Mahony, R., and Scaramuzza, D. (2019). CED: color event camera dataset. *arXiv preprint arXiv:1904.10772*. doi: 10.1109/CVPRW.2019.00215

Shiba, S., Aoki, Y., and Gallego, G. (2022). "Secrets of event-based optical flow," in *Computer Vision–ECCV 2022: 17th European Conference* (Tel Aviv: Springer), 628–645.

Ye, C., Mitrokhin, A., Fermüller, C., Yorke, J. A., and Aloimonos, Y. (2020). "Unsupervised learning of dense optical flow, depth and egomotion with event-based sensors," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE), 5831–5838. doi: 10.1109/IROS45743.2020.9341224

Yu, J. J., Harley, A. W., and Derpanis, K. G. (2016). Back to basics: unsupervised learning of optical flow via brightness constancy and motion smoothness. *arXiv preprint arXiv:1608.05842*. doi: 10.1007/978-3-319-49409-8_1

Zhang, K., Che, K., Zhang, J., Cheng, J., Zhang, Z., Guo, Q., et al. (2022). "Discrete time convolution for fast event-based stereo," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (New Orleans, LA: IEEE), 8666–8676. doi: 10.1109/CVPR52688.2022.00848

Zhang, Y., Lv, H., Zhao, Y., Feng, Y., Liu, H., and Bi, G. (2023). Event-based optical flow estimation with spatio-temporal backpropagation trained spiking neural network. *Micromachines* 14, 203. doi: 10.3390/mi14010203

Zhu, A. Z., Thakur, D., Ozaslan, T., Pfrommer, B., Kumar, V., and Daniilidis, K. (2018a). The multivehicle stereo event camera dataset: an event camera dataset for 3d perception. *arXiv preprint arXiv:1801.10202*. doi: 10.1109/LRA.2018.2800793

Zhu, A. Z., Yuan, L., Chaney, K., and Daniilidis, K. (2018b). EV-flowNet: self-supervised optical flow estimation for event-based cameras. *arXiv preprint arXiv:1802.06898*. doi: 10.15607/RSS.2018.XIV.062

Zhu, A. Z., Yuan, L., Chaney, K., and Daniilidis, K. (2019). "Unsupervised event-based learning of optical flow, depth, and egomotion," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, CA: IEEE), 988–997. doi: 10.1109/CVPR.2019.00108