Optical Flow Determination using Neuromorphic Hardware with Integrate & Fire Neurons

Francesco Branca*
Micro Air Vehicle Laboratory
Delft University of Technology
Delft, Netherlands
F.Branca@student.tudelft.nl

Jesse Hagenaars†
Micro Air Vehicle Laboratory
Delft University of Technology
Delft, Netherlands
J.J.Hagenaars@tudelft.nl

Guido de Croon†
Micro Air Vehicle Laboratory
Delft University of Technology
Delft, Netherlands
G.C.H.E.deCroon@tudelft.nl

* MSc Student, † Supervisor

***Abstract*—Spiking neural networks implemented for sensing and control of robots have the potential to achieve lower latency and power consumption by processing information sparsely and asynchronously. They have been used on neuromorphic devices to estimate optical flow for micro air vehicles navigation, however robotic implementations have been limited to hardware setups with sensing and processing as separate systems. This article investigates a new approach for training a spiking neural network for optical flow to be deployed on the speck2e device from Synsense. The method takes into account the restrictions of the speck2e in terms of network architecture, neuron model, and number of synaptic operations and it involves training a recurrent neural network with ReLU activation functions, which is subsequently converted into a spiking network. A system of weight rescaling is applied after conversion, to ensure optimal information flow between the layers. Our study shows that it is possible to estimate optical flow with Integrate-and-Fire neurons. However, currently, the optical flow estimation performance is still hampered by the number of synaptic operations. As a result, the network presented in this work is able to estimate optical flow in a range of [-4, 1] pixel/s.**

***Index Terms*—neuromorphic computing, spiking neural networks, integrate-and-fire, computer vision, optical flow, micro air vehicles**

I. Introduction

Neuromorphic computing has emerged as a promising paradigm for biologically inspired robotics applications [1, 2, 3, 4]. In contrast to von-Neumann architectures, the information processing in neuromorphic devices occurs asynchronously and sparsely. The networks hosted on these type of processors are called spiking neural networks (SNNs) [5], also known as the third generation of neural networks. SNNs contain layers of neurons communicating with each other through binary inputs called "spikes" [6]. Similarly to the structure and functionality of the human brain, SNNs are able to perform parallel computations. When implemented on neuromorphic hardware, SNNs result in lower power consumption and latency, which is preferable for efficient real-time systems.

Neuromorphic devices have been used for a variety of applications in computer vision, ranging from object detection [7], tracking [8, 9] and gesture recognition [10, 11]. Regarding time-dependent tasks, one application is determining optical flow for Micro Air Vehicles (MAVs) state estimation [12]. Optical flow is a concept in computer vision that quantifies the apparent motion of points in an image [13]. It was discovered that optical flow serves as an important visual cue for animals. For instance, bees exploit optical flow to land by keeping the image velocity constant [14]. Other studies also hypothesized that it is also used to estimate distances and avoid obstacles [15, 16].

Determining optical flow using SNNs can be done by combining neuromorphic devices with event-based cameras (EBCs) [17]. This type of vision sensors are composed by an array of pixels, where each pixel detects changes in brightness in the image and outputs a binary variable indicating a positive or negative change. Since the static elements in the image are not detected, event-based vision sensors avoid capturing redundant data and result in extremely low power consumption. EBCs consume on average 10 mW of power, while standard cameras power consumption is in the scale of W [17]. This makes them suitable for real-time sensing in robotics systems.

With faster and more energy efficient computations, MAVs can be designed to be lighter, more agile and more similar to real insects. This technology could be helpful in applications such as search and rescue and flying in greenhouses to monitor crop. Paredes-Vallés et al. [18] proposed the first fully neuromorphic vision-to-control pipeline for controlling a MAV. The system runs completely autonomously with the help of Intel's Loihi [19] processor and a DAVIS240C event-based camera.

This article presents a novel approach to estimating optical flow, suited for implementation on lighter and more simple neuromorphic devices. The chip used for experiments is Synsense's speck2e [20, 21, 22], a dynamic vision system-on-chip, integrating sensing and computing in one board. The speck2e is equipped with Integrate-and-Fire neurons. To achieve a performance comparable to the Intel's Loihi, a new method has to be developed to transmit information inside the network, without relying on the leakage system. The prime goal of this research is to design a speck2e-compatible network that is able to estimate direction and magnitude of dense optical flow from an event-based dataset. A Recurrent Neural Network (RNN) with ReLU activation functions is trained in self-supervised learning and converted to spiking for testing. The converted network is then deployed on the speck2e to assess the hardware performance.

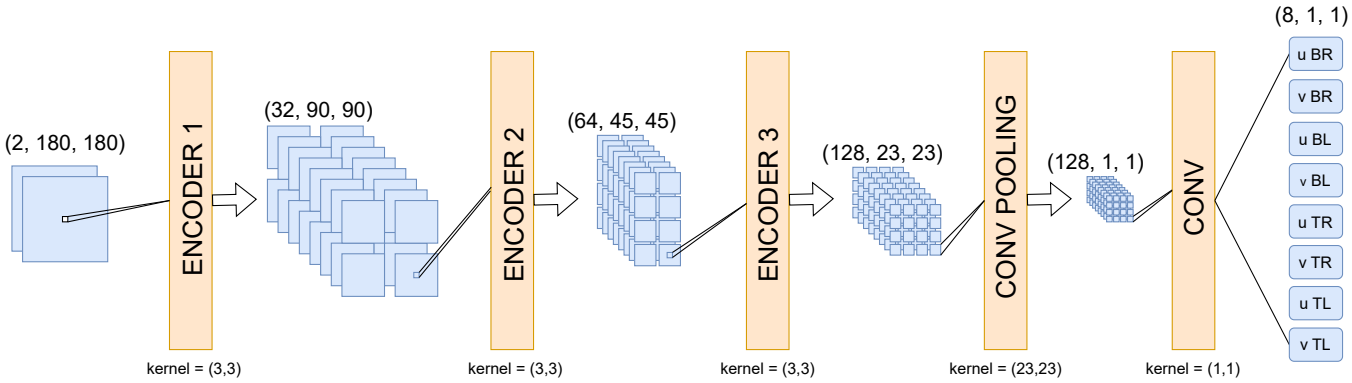


Fig. 1. LIF network diagram from Paredes-Vallés et al. [18].

II. Related Work

A. Neuromorphic Robotics

Neuromorphic devices have been previously used for real-time control of robotics systems. For instance, neuromorphic systems have been sued for control of particular robotic joints to achieve a certain motion [23, 24, 25]. In the field of computer vision, there has been extensive research into developing systems with autonomous navigation using obstacle avoidance and edge tracking [26, 27, 28, 29].

B. Neuromorphic Control of Micro Air Vehicles

Regarding autonomous navigation of MAVs, previous works have implemented neuromorphic computing for sensing and controlling. Hagenaars et al. [30] used SNNs for the first time to control a drone during landing, using optical flow divergence. Following, Paredes-Vallés et al. [18] introduced the first fully neuromorphic pipeline for drone control. A SNN was used to process events from a camera and output low-level control actions to perform autonomous vision-based flight. Both of these works used neuromorphic hardware with fully programmable LIF neuron models. The drone setup included a DAVIS240C event-based camera, a Loihi neuromorphic chip [19] and a a single-board computer UP Squared. The latter component was used to pre-process the events (downsampling and cropping), as well as process the output spikes from the Loihi. A further improvement would be implementing a new device that combines vision and processing in the same board, without requiring an additional component in between.

The architecture proposed in Paredes-Vallés et al. [18] (Figure 1), is denoted here as LIF-3 and it includes three groups of layers (encoders) with recurrent connections and LIF spiking activation functions, one pooling layer and one prediction layer. In total, 4 networks were used, one for each corner of the image and each corner was cropped to a smaller region of interest, in order to reduce the number of input events. Note that defining 4 different regions of interest is not possible with the DVS on the speck2e. To fairly compare the speck2e architectures to LIF-3, the same number of channels per layer is used, however the network is applied to the full 180x180 picture instead of 4 different corners. Note that the output of the LIF-3 is still 8 flow vector components, as shown in Figure 1. In this article, this architecture is used as a starting point for the development of a new network configuration.

C. Previous Speck2e Implementations

Other examples of computer vision tasks tested on the speck2e are face detection from Caccavella et al. [7], binary particle classification [31], CIFAR10, ImageNet and NMNIST classification [32, 33]. This neuromorphic device has not been used yet for time-dependent tasks, requiring recurrent connections for integrating in time.

III. Methodology

A. Event Based Cameras

Event-based cameras react to changes in brightness in the scene by providing as output a stream of independent and sparse inputs. Each event is defined as $\mathbf{e_i} = (x_i, y_i, t_i, p_i)$, where x_i, y_i are the coordinates of the pixel, t_i is the timestamp in microseconds and $p_i \in [-1, 1]$ is the polarity of the event (-1 for decrease and $+1$ for increase in brightness). The asynchronous nature of EBCs is suitable for on board processing on a neuromorphic device. For instance, the speck2e has a processing frequency of 1 MHz, hence a resolution in time of microseconds order. However, when training the SNNs in simulation, the processing frequency has to be decreased in order to speed up the training time, since processing each event individually would take too long. For this reason, the events are binned into event frames, which can be more easily processed by deep learning frameworks [17]. In the following experiments, when training the network, the events are accumulated in time windows of 5 ms.

B. Optical Flow Model

The network architecture proposed estimates dense (per pixel) optical flow in horizontal and vertical direction (u, v) in pixels per millisecond. The network takes the input from the whole DVS array of pixels and predicts the flow for 4 sections of the image, hence it outputs a total of 8 predictions. The corners are denoted as top left (TL), top right (TR), bottom left (BL) and bottom right (BR) (Figure 1). Note that the flow

is two-dimensional, hence it is assumed that the EBCs used to record the camera of the dataset is looking at a planar surface.

C. Hardware Constraints

The speck2e circuit is able to support large-scale SNN for various computer vision tasks, such as sign recognition, smart tracking and obstacle detection. The final network configuration shall be able to estimate optical flow from the datasets accurately and the architecture shall be designed to fit the hardware constraints. The speck2e specifications are [34]:

1) Maximum of 9 convolutional layers
2) Maximum number of neurons: 32k
3) Maximum input dimension: 128x128
4) Maximum feature output size: 64x64
5) Maximum feature number: 1024
6) Weight resolution: 8 bit
7) Neuron state resolution: 16 bit
8) Maximum kernel size: 16x16
9) Stride: {1, 2, 4, 8} independent in X/Y
10) Padding: [0...7] independent in X/Y
11) Pooling: 1:1, 1:2, 1:4
12) Fanout: 2
13) Frequency: 1 MHz
14) Synaptic Operations limit per core: 10 millions synops/s
15) Maximum number of channels in readout layer: 15

Importantly, the Speck2e features an Integrate-and-Fire (IF) model. In comparison with a Leak-Integrate-and-Fire (LIF) model, the IF model is simpler and it requires less energy. However, it is also more limited, and at the onset of our study it was unclear if an IF model SNN would be able to estimate optical flow. In the next subsection, we explain the differences between LIF and IF model in more detail.

D. Neuron Model Comparison

This section describes the differences between the LIF model available on the Loihi chip and the IF model from the speck2e. Each layer in a SNN can be visualized as a convolutional layer plus a spiking activation function. The layer contains weights and biases, which define how much the input influences the potential of the neuron and therefore its output. The spiking function outputs a 1 or a 0 at every step, depending on whether the potential exceeded the threshold or not. A schematic of the neurons is provided in Figure 2 for LIF and Figure 3 for IF.

$$s(v[t]) = \begin{cases} 0 & \text{if } v[t] < v_{\text{mem}} \\ 1 & \text{if } v[t] > v_{\text{mem}} \end{cases} \tag{1}$$

$$\begin{aligned} i[t+1] &= w \cdot z[t] + \theta_i \cdot i[t] + b \\ v[t+1] &= v[t] \cdot \theta_v \cdot (1 - s(v[t])) + i[t+1] \end{aligned} \tag{2}$$

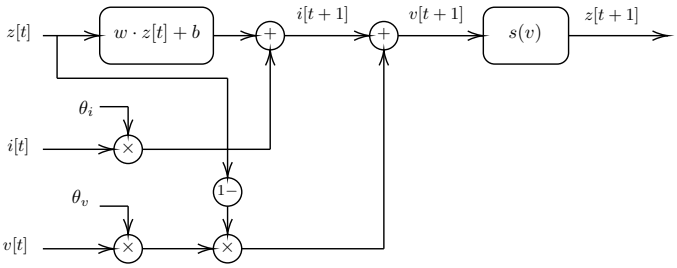


Fig. 2. LIF neuron computational graph.

$z(t)$ is the input spikes vector, $i(t)$ and $v(t)$ are the neurons' current and potential values. Additionally, the potential is multiplied by a reset term $(1 - z)$, so that if a neuron spiked in the previous step ($z = 1$), the potential is reset to zero. The spikes are fed in the convolutional layer, where they are multiplied by the weights and summed with the biases. The output of the convolutional layer represents the change in current due to the spikes, which is added to the previous state current. The new current $i(t+1)$ is summed with the previous potential to compute the new potential $v(t + 1)$. Finally, the spiking function $z(v)$ checks if the potential is higher than the threshold and transmits spikes to the following layer.

In the LIF model, θ_i and θ_v stand for the current and potential leak respectively and they are multiplied by the previous state values. The leak represents how much information of the previous state is preserved in the following step. A higher leak value means that the current or potential decreases more slowly and more information is maintained through time. In the IF model the leak parameters are not present and the new incoming current is directly summed to the previous one.

$$\begin{aligned} i[t+1] &= w \cdot z[t] + i[t] + b \\ v[t+1] &= v[t] \cdot (1 - s(v)) + i[t+1] \end{aligned} \tag{3}$$

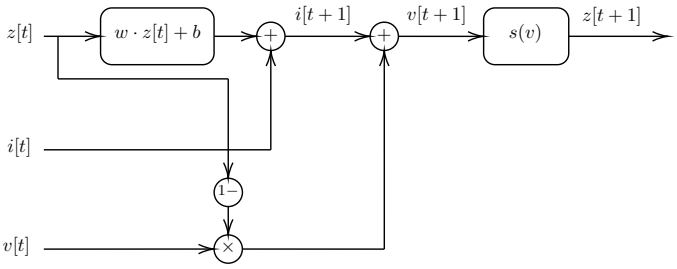


Fig. 3. IF neuron computational graph.

E. Sinabs Neuron Model

Sinabs [33] is a deep learning library based on PyTorch provided by Synsense. It is suited to test SNNs to be implemented on Synsense devices such as the speck2e. The sinabs model of an IF spiking layer is denoted as `IAFSqueeze` and it is characterized by the following parameters:

- `spike_fn` - Function defining the spiking output. If set to `MultiSpike`, a neuron will be able to produce mul-

tiple spikes in given time step. If set to `SingleSpike`, a neuron will produce at most one spike per time step.
- `reset_fn` - Function defining the reset phase of the potential. If set to `MembraneSubtract`, the threshold value is subtracted from the potential after spiking. If set to `MembraneReset`, the potential returns to zero after spiking.

For testing the network in simulation, the `IAFSqueeze` functions are set to `MultiSpike` mode. If the neurons in a certain binned frame are able to send multiple spikes, they can encode more information in that specific time and the flow can be better characterized. Regarding the reset function, the `MembraneReset` option is chosen instead of `MembraneSubtract`, in order to avoid membrane potential accumulation.

F. Recurrent Connections

For time-dependent tasks, IF SNNs would require recurrent connections in order to make predictions on the current state, while considering the previous one. There are two possible recurrent encoder configurations on the speck2e.

Sum Recurrency

In this type of encoder the first layer increases the number of channels from n_1 to n_2. The second layer processes the new input from the first layer, together with the previous state. The output of the second forward layer is summed directly to the new incoming state and then fed in the layer.

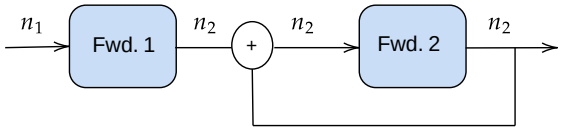


Fig. 4. Sum Recurrency

Concatenation Recurrency

This type of encoder has two forward layers and one recurrent layer. The first forward layer increases the number of channels from n_1 to n_2. The second forward layer processes the output of the first forward layer concatenated with the one of the recurrent layer, hence it has $2 \cdot n_2$ inputs and $2 \cdot n_2$ output channels. Finally, the recurrent convolutional layer decreases the number of channels back from $2 \cdot n_2$ to n_2.

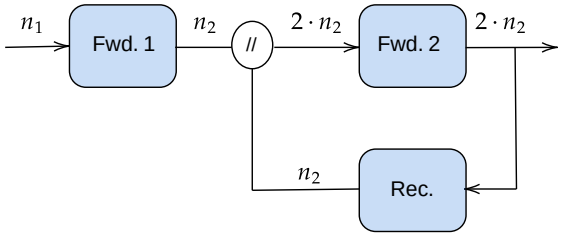


Fig. 5. Concatenation recurrency diagram.

G. Network Conversion & Weight rescaling

The Speck2e has been successfully used for recognition tasks (Section II). However, determining optical flow requires more extensive temporal processing of the events. This is because the network has to integrate the information of the previous states in time to determine the current state. With IF neurons, all the valuable information among layers is encoded in the spikes and the recurrent connections are the only memory mechanism in every layer.

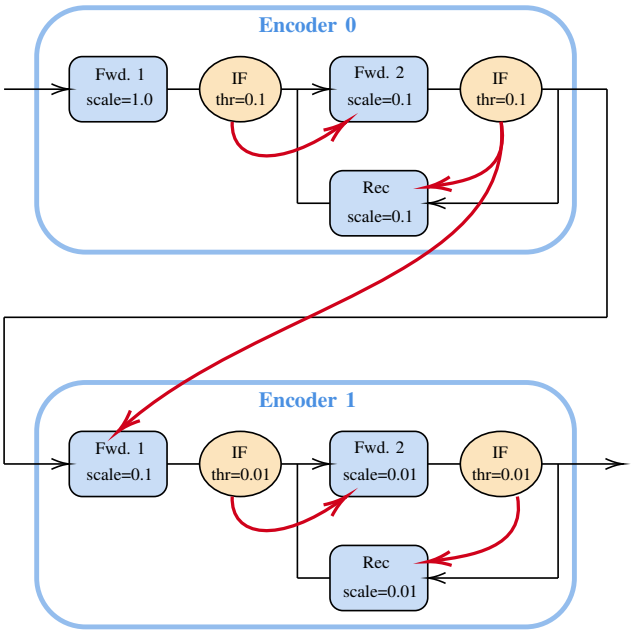


Fig. 6. Weight rescaling diagram.

A common strategy to train SNNs is to first train an ANN for ReLU activation functions and then replace them with spiking activation functions, as explained in Cao et al. [35]. This method works well for non time-dependent tasks such as image classification and object detection [10, 7]. In sinabs, when substituting ReLU functions with `IAFSqueeze`, the thresholds are set to $\pm$ 1.0 by default. However, if the synaptic weights have a different range of values, this thresholds might not be suitable and the layer might result in too high or too low activity. Rueckauer et al. [36] proposes a solution to this problem, which consists in normalizing the activity in every layer by scaling the weights to be in the appropriate range. However, this method was only applied to an object detection task, hence without recurrent connections and time dependency. In this article we propose the same approach as in Rueckauer et al.[36], but for a recurrent spiking network. The main difference is that, while for recognition tasks the weights are scaled to be in the right rang but still lower than the thresholds, in our method the weights are required to be higher than the thresholds.

Without weight rescaling, the information in every layer tends to accumulate membrane potential, instead of transmitting spikes to the following layers. If the information between

layers is not promptly transmitted as it arrives, the recurrent connections are not going to be activated and therefore the network will lose the previous states. Setting the threshold too high with respect to the scale of the synaptic inputs will result in losing relevant information over time, while setting it too low will result in excessive information flow.

By observing the range of the output of the ReLU functions in every layer, a common threshold can be defined for every encoder. The threshold should be low enough to allow some spikes to pass through at every binned frame of 5 ms. Note that finding the optimal threshold value to allow the minimum number of spikes to pass, while still encoding the relevant information, is a complex optimization problem. There is not one ideal scaling factor for the weights, as this will depend on many parameters such as the spike rate in the dataset, the network depth, the spike threshold and many more. For this reason the thresholds will be selected empirically.

In the first layer of the first encoder, the ReLU outputs are in the range 10^{-1}, thus the upper and lower thresholds are set to $\pm$ 0.1 for all layers of the first encoder. The weights of the following layer are then re-scaled by 0.1. This pattern of rescaling the weights by the threshold value of the previous layer is repeated for all layers, as shown in Figure 6. In the second encoder the synaptic output range is in the order of 10^{-2}, thus the threshold is set to $\pm$ 0.01.

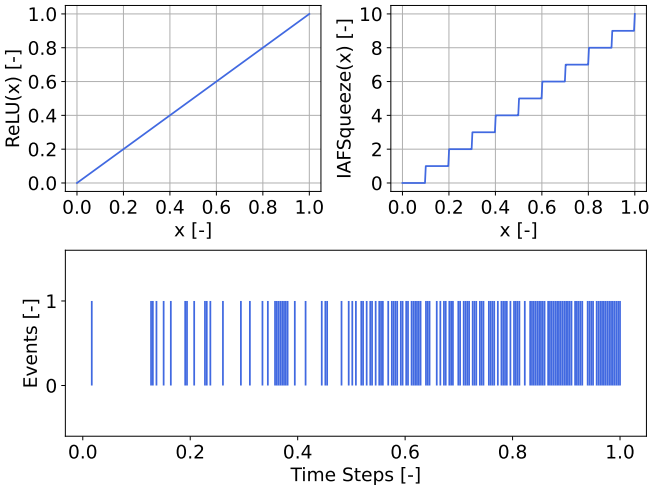


Fig. 7. Activation functions and resulting spike rate output.

Figure 7 shows the output of the ReLU function (continuous) compared to the output of the `IAFSqueeze` function (discrete) and the resulting spike rate. The idea of ANN-to-SNN conversion is using only the spikes to transmit information through layers and therefore reducing the resolution or sampling the information. Equation 4 shows that the output of the IF function z_{IF} is equal to the integer part of the ReLU output z_{ReLU} divided by the threshold v_{mem}. For instance, if the input of a neuron equal to 0.421 and the threshold of the neuron is set to 0.1, 4 spikes will be transmitted to all the neighboring neurons.

$$z_{\text{IF}} = \left\lfloor \frac{z_{\text{ReLU}}}{v_{\text{mem}}} \right\rfloor \tag{4}$$

H. Self-Supervised Loss Function

The network is trained in self-supervised learning. The loss function for is composed by two terms: *contrast maximization* and *flow smoothing*.

1) Contrast Maximization: this function quantifies the flow loss given a set of events. It was proposed by Gallego et al. [37] and it solely relies on the events stream, without any additional data required. Consider a set of positive and negative events in a certain spatio-temporal neighborhood. The events triggered by the same moving edges are expected to follow the same trajectories. The translational displacement of the pixels can be described by Equation 5, where (u, v) are the flow components.

$$\begin{pmatrix} x_i' \\ y_i' \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} + (t - t_i') \begin{pmatrix} u(x_i, y_i) \\ v(x_i, y_i) \end{pmatrix} \tag{5}$$

Calculating the correct flow can be interpreted as finding the best fitting trajectory that passes through the events generated by the same moving edge. Consider the scenario displayed in Figure 8. During training the network performs 5 forward passes before computing the loss. For each of the 5 frames, the events are transposed in time using the last flow vectors estimation. Essentially, the 5 frames are used to reconstruct the events at a reference time. The 5 reconstructions are then summed together and, if the flow estimation is correct, the pixels should be aligning without any blur. In reality, the first optical flow estimates will not be correct and will therefore result in a blurred reconstructed image (Figure 8). In order to measure this blur and minimize it, the density of events per pixels is calculated. If the flow estimation is accurate, the reconstructed events will be aligning on the same pixel.

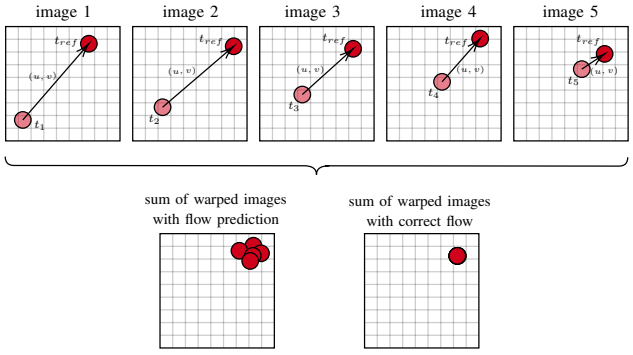


Fig. 8. Contrast maximization computation scheme.

To compute the contrast maximization loss function, first the events are separated by polarity. Following, the average timestamp image is generated at each pixel for each polarity, as in Zhu et al. [38].

$$T_{p'}(\mathbf{x};\mathbf{u}|t_{\text{ref}}) = \frac{\sum_j \kappa(x-x'_j)\kappa(y-y'_j)t_j}{\sum_j \kappa(x-x'_j)\kappa(y-y'_j)+\epsilon} \qquad \kappa(a) = \max(0, 1-|a|) \tag{6}$$

To make the loss function convex, in Paredes-Vallés et al. [39] the contrast maximization function was scaled with the number of pixels with at least one warped event (Equation 7).

$$\ell_{\text{contrast}}(t_{\text{ref}}) = \frac{\sum_{\mathbf{x};\mathbf{u}} T_+(\mathbf{x};\mathbf{u}|t_{\text{ref}})^2 + T_-(\mathbf{x};\mathbf{u}|t_{\text{ref}})^2}{\sum_{\mathbf{x}}[n(\mathbf{x}')>0]+\epsilon} \tag{7}$$

2) Flow Smoothing: this term encourages smoothness in the estimated optical flow field. It is based on the assumption that neighboring pixels should have similar flow values. The smoothness function is meant to regularize the output flow and it is applied in the temporal domain to subsequent per-corner optical flow estimates.

$$\ell_{\text{smooth}} = \sum_{x,y} \sum_{i,j\in\mathcal{N}(x,y)} \rho(u(x,y)-u(i,j))+\rho(v(x,y)-v(i,j)) \tag{8}$$

The total loss function used during training to learn optical flow is Equation 9, where λ is a scalar balancing the two functions.

$$L_{\text{total}} = \ell_{\text{contrast}} + \lambda \ell_{\text{smooth}} \tag{9}$$

I. Synops Loss Function

As specified in Section III-C, the limit on the number of synaptic operations is 10 millions synops/s. By synaptic operation, it is meant all the spikes sent from the neurons of one layer to the neurons of another layer. In order to regularize the activity of the layers during training and prevent excessive spiking, Equation 10 is introduced as an additional term in the loss function.

$$\text{loss} = \sum_{k=1}^{N} \frac{(\text{total output})_k}{(\text{threshold})_k (\#\text{ parameters})_k} \tag{10}$$

The total output of every layer at every step is the total number of spikes that a layer produces. During training the network is using ReLU functions and the output of every neuron is a floating point. However the range of weights values for every layer changes and, if it is not rescaled, the deeper layers with lower weight values will have less importance in the cost function. Moreover, layers with more parameters need to have a higher spike rate to send more information, thus the layer output is also scaled by the number of parameters. The weight of the synops loss term on the total loss function has to be tuned, in order to achieve a balance that allows to learn optical flow properly and with the minimum number of operations.

IV. Simulation Results

A. Training and Testing Datasets

The network is trained on two main datasets. The first one is addressed as CyberZoo dataset[18] and it includes 40 minutes of event data, which are split into 25 minutes for training and 15 for testing. This dataset includes translational and rotational motion in multiple directions and at different speeds. The CyberZoo dataset will be used to quantify the network's accuracy with respect to ground truth data and assess its performance.

The second dataset is provided by the University of Zurich and it is denoted as Davis dataset [40]. This dataset includes two main video sequences of 60 seconds each. It contains translational and rotational motion, however due to its limited size, it is not as complex and diversified as the CyberZoo dataset. It was observed that networks trained on the Cyberzoo dataset have a higher number of synops with respect to networks trained on the Davis dataset. This is most likely due to the higher complexity of the Cyberzoo sequences, which require more spikes to identify the motion. Therefore, for the hardware experiments, a sequence from the Davis dataset is used as input in the speck2e, instead of the events coming from the DVS. This is done primarily to avoid collecting another dataset and to validate the network on more simple motions.

B. Network Configurations

A total of 5 networks is tested on the datasets and compared to the peformance of the LIF network used in Paredes-Vallés et al. [18].

1) RNN-2-S: recurrent neural network with 2 encoders, sum recurrency and ReLU activation functions.
2) IF-2-S: spiking equivalent of RNN-2-S.
3) RNN-2-C: recurrent neural network with 2 encoders, concatenation recurrency and ReLU activation functions.
4) IF-2-C: spiking equivalent of RNN-2-C.

Note that the number of encoders is decreased from 3 to 2, because of the limit on number of convolutional cores on the speck2e.

C. Results on CyberZoo dataset

The network architectures are tested on the CyberZoo dataset and compared to the LIF network. The comparison metrics are Signal-to-Noise Ratio (SNR), Average Endpoint Error (AEE) and The Ratio of Squared Average Timestamps (RSAT). Table I and Figure 9 shows the results for the different network configurations compared to the LIF-3 [18]. All the metrics are computed on a single sequence from the dataset.

Signal-to-Noise Ratio

To calculate the SNR, a sequence of 5.0 s from the testing dataset is considered. The network outputs 8 optical flow predictions, a horizontal and a vertical component for each

of the 4 corners of the image. To distinguish the power of the signal from the noise, the predictions are transposed to the frequency domain using Fast Fourier Transform (FFT).

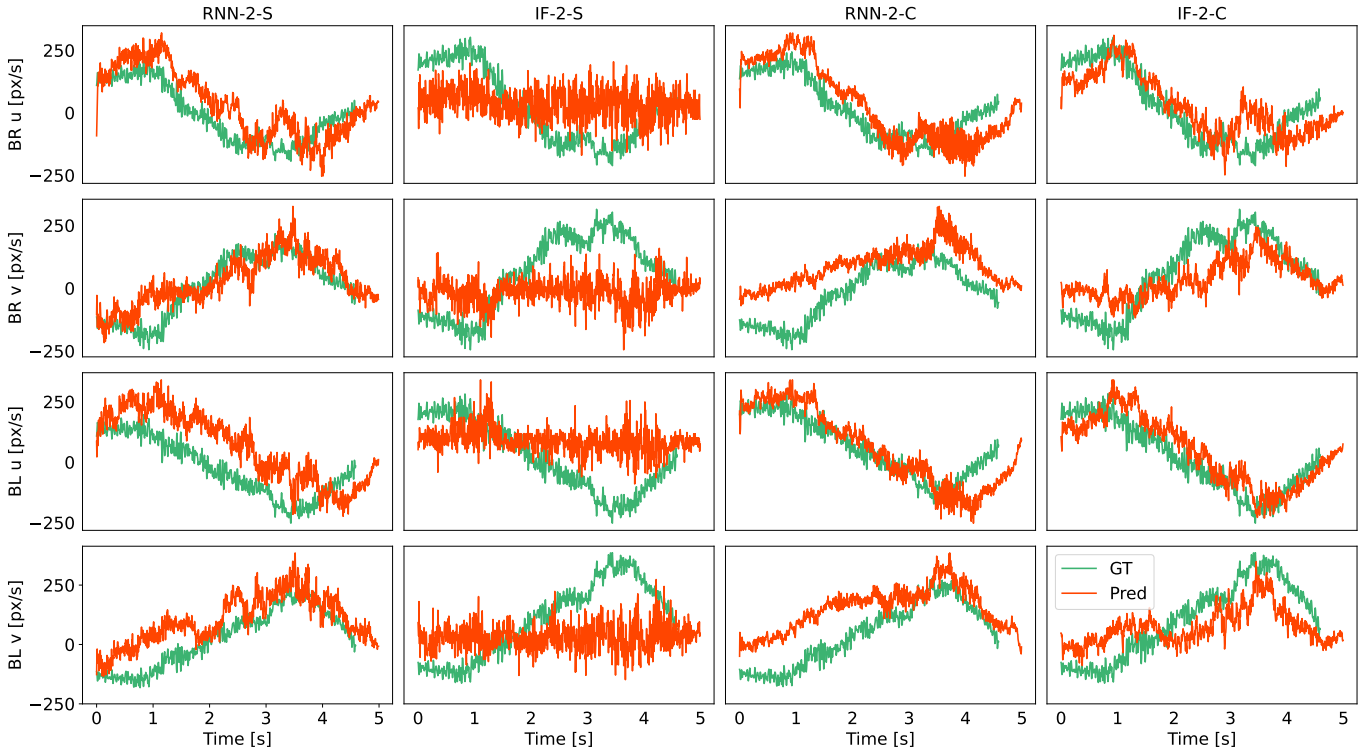


Fig. 9. Ground truth compared to network prediction for RNN-2-S, IF-2-S, RNN-2-C, IF-2-C

In this way, the main signal frequency can be identified by observing the peaks. The frequency range of the main signal is defined by observing the FFT of the ground truth signal. Using Equation 11 the average SNR can be calculated for each flow vector signal and compared to the ground truth.

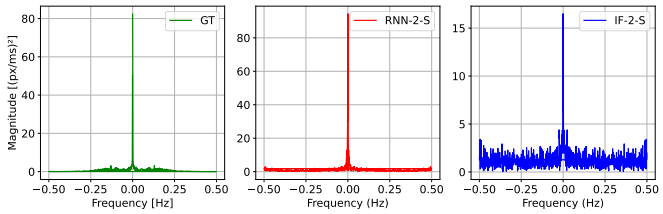


Fig. 10. FFT of ground truth signal compared to RNN-2-S and IF-2-S.

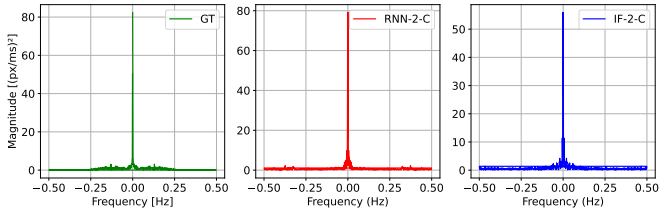


Fig. 11. FFT of ground truth signal compared to RNN-2-C and IF-2-C.

Figure 10 and 11 show the FFT of the ground truth signal compared to the network predictions of the ReLU and IF network. By zooming in the graph, it is found that the ground truth has most of its power in the frequencies $\pm 2.5 \cdot 10^{-3}$ Hz, hence this range is chosen to define the main signal. The FFT shows that the IF prediction is weaker in that range and has more power in other frequencies, hence the SNR is expected to be lower.

$$\text{SNR} = 10 \cdot \log_{10}\left(\frac{P_{\text{signal}}}{P_{\text{noise}}}\right) \tag{11}$$

Average Endpoint Error & Standard Deviation

The Average Endpoint Error (AEE) is a metric used to evaluate the accuracy of optical flow algorithms. It is calculated by taking the each prediction of the network per corner of the image (D) and calculating the squared difference with ground truth. Then the respective errors of u and v are summed and the average of all corners is taken. This value represents the endpoint error at a certain time step. For all the error values the average and standard deviation can be calculated. The AEE is the average of all endpoint errors at every time step (N)

$$e_i = \frac{1}{D}\sum_{j}^{D}\left[(u_j - \hat{u}_j)^2 + (v_j - \hat{v}_j)^2\right] \tag{12}$$

$$\text{AEE} = \frac{1}{N}\sum_{i}^{N} e_i \quad \text{STD} = \sqrt{\frac{1}{N}\sum_{i}^{N}(e_i - \text{AEE})^2} \tag{13}$$

Ratio of Squared Average Timestamps

RSAT is the ratio of the squared sum of the per-pixel and

per-polarity average timestamp of the image of warped events and that of the image of (non-warped) events. Essentially, the RSAT measures the sharpness of the reconstructed image and it is an indication of how well the flow is estimated. The lower the value of this metric, the better the optical flow estimate. Note that this metric is sensitive to the number of input events.

TABLE I
SNR, AEE AND RSAT FOR THE DIFFERENT NETWORKS ON A CYBERZOO DATASET SEQUENCE.

Configuration	SNR (dB) ↑	AEE ↓	STD ↓	RSAT ↓
LIF-3 [18]	9.09	0.199	0.0531	0.917
RNN-2-S	9.11	0.183	0.0585	0.913
IF-2-S	-0.877	0.163	0.0637	0.987
RNN-2-C	12.2	0.211	0.0567	0.896
IF-2-C	6.98	0.171	0.0668	0.965

D. Network Configuration Trade-off

In this section, the results of the different network configurations on the Cyberzoo dataset are discussed.

Recurrency Type

With the same number of encoders, RNN-2-S has a slightly lower AEE than RNN-2-C and a more or less similar STD, although this does not necessarily indicate a better performance, since the RSAT is lower for concatenation networks. By observing the predictions compared to ground truth in Figure 9 it can be seen that both networks are able to follow the motion fairly accurately. However, in terms of noisiness, RNN-2-C and IF-2-C have a better SNR. In Figure 9 it can be seen that RNN-2-S predictions are more subject to additional fluctuations that are not present in the ground truth signal. Moreover, when converted, IF-2-S does not work properly with the weight rescaling method and most of the main signal seems to be lost. Concatenation seems to work better in this case, most likely because of its additional convolutional layer in the recurrent connection. This layer processes the previous state and it serves as a filter for non-relevant information. With sum recurrency, the previous state is directly summed with the new incoming state and then processed, therefore the convolutional layer does not distinguish between past and present information.

Network Conversion

Converting RNN to IF inevitably lowers the SNR and introduces additional noisy frequencies. When the network is converted, the resolution of the information between layers is reduced and therefore output signal becomes less defined. With sum recurrency (SNR = -0.877 dB), the conversion has way more impact on the SNR than with concatenation (SNR = 6.98 dB). Moreover, when converting, the AEE drops while the RSAT and STD increase. As specified earlier, a lower AEE does not indicate better performance. Indeed the STD increase proves that the errors are more dispersed.

Neuron Models

From the previous discussion, it was established that IF-2-C appears to be a more suitable option. Compared to LIF-3, IF-2-C performance worse in most of the metrics. This is because LIF-3 not only has more hyperparameters than IF-2-C, but also a higher level of complexity of the neuron model. LIF neurons are better at capturing temporal dynamics, since not all the information has to be sent through the spikes and some of the irrelevant one is filtered out by the leak system. Despite the limitations, the IF-2-C is able to achieve a comparable performance with respect to LIF-3.

E. Results on Davis Dataset

The Davis dataset does not include ground truth optical flow, thus the performance of the network is evaluated by visual inspection. The size of the dataset sequences is reduced from 128 pixels to 90, in order to further reduce the number of synaptic operations for hardware implementation. Figure 12 shows the output of IF-2-C on a horizontal and a vertical motion sequence. The pictures on top represent the color-coded dense optical flow. Every pixel has a different color depending on the flow at that point (check Figure 19 for color map).

F. Synaptic Operations Analysis

Besides using the synops loss term, some of the network hyperparameters can be altered to influence the number of synops.

Stride

To minimize the synops, the stride of the first encoder can be changed to 4, as in the second encoder. By halving the size of the images in the first encoder, less information is forwarded, hence the layers produce less synops (Figure 20 and 21).

Number of Channels

Increasing the number of channels can make the network's activity more sparse. Choosing a configuration with more neurons means having more neurons doing less work, therefore less operations per neuron overall (Figure 22 and 23).

Early Stop

During training, the network establishes the synaptic weights to optimize for minimal loss. However, in this case the minimum loss is often achieved in the first few epochs. By letting the network train more, additional connections are formed, which make the predictions more accurate. These redundant weights are useful for making the network more precise, however they increase the number of operations significantly. If the training is stopped earlier and the model is still able to estimate the flow accurately, a more operations efficient result can be obtained (Figure 24 and 25).

Considering the influence of the hyperparameters, a new network configuration is trained for implementation on the speck2e (Table IV).

G. Synops Loss Term

Recurrent connections on the speck2e process at a frequency of 1 MHz, thus the number of synops could change significantly compared to IF-2-C in simulation. Nevertheless, the total number of spikes produced by all the layers can still be reduced by including the synops loss term explained in Section III-I.

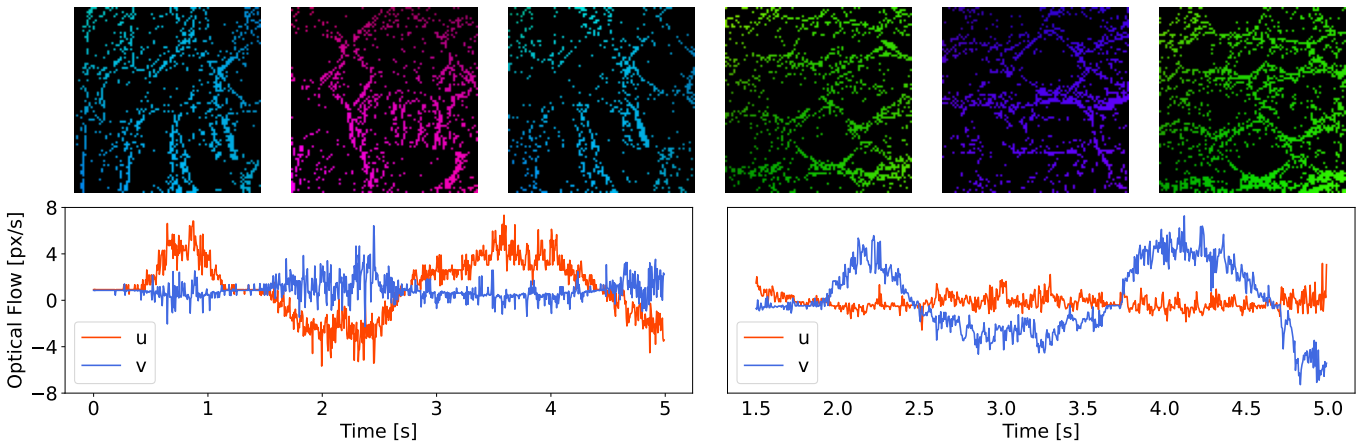


Fig. 12. IF-2-C optical flow vectors for the Davis dataset sequences.

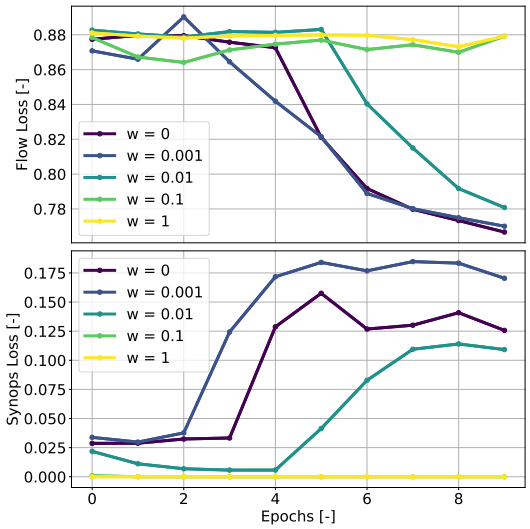


Fig. 13. Flow and synops loss during training with different weighting.

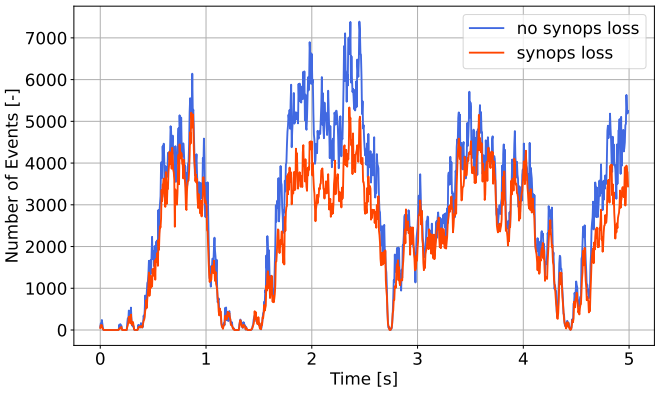


Fig. 14. Total number of synops for network trained with synops loss compared to network trained without synops loss.

Figure 13 shows the flow loss and synops loss over 10 epochs. By looking at the synops loss, it is clear that 0.01 is an optimal value for the weight, as it allows the flow loss to converge to a slightly higher value, while keeping the synops lower. If the weight is increased to 0.1, the network minimizes the synops to zero and it is prevented from learning flow. If it is decreased to 0.001, its influence is almost unnoticeable and the synops increase to more than necessary. Figure 14 shows the influence of the loss function on the total number of spikes of all the layers in the network.

V. Hardware Implementation

A. Hardware Setup Overview

The speck2e hosts the 2 encoders and the pooling layer, while the prediction layer post-processes the spikes outside the chip and translate them into optical flow vectors. The spikes are binned in time windows of 5 ms before being fed into the prediction layer. For the preliminary tests, the input is not from the DVS, but instead a sequence from the Davis dataset is used. A diagram of the hardware setup is shown in Figure 15.

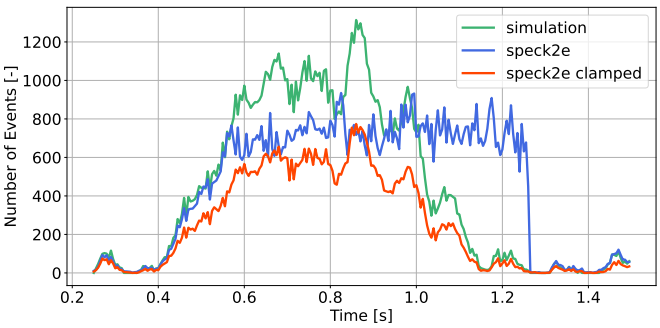


Fig. 16. Spike activity comparison

B. Spike Activity Comparison

Figure 16 shows the total number of spikes for every 5 ms frame in the first layer of the first encoder. While in simulation the layer follows a specific trend dictated by the input spikes,

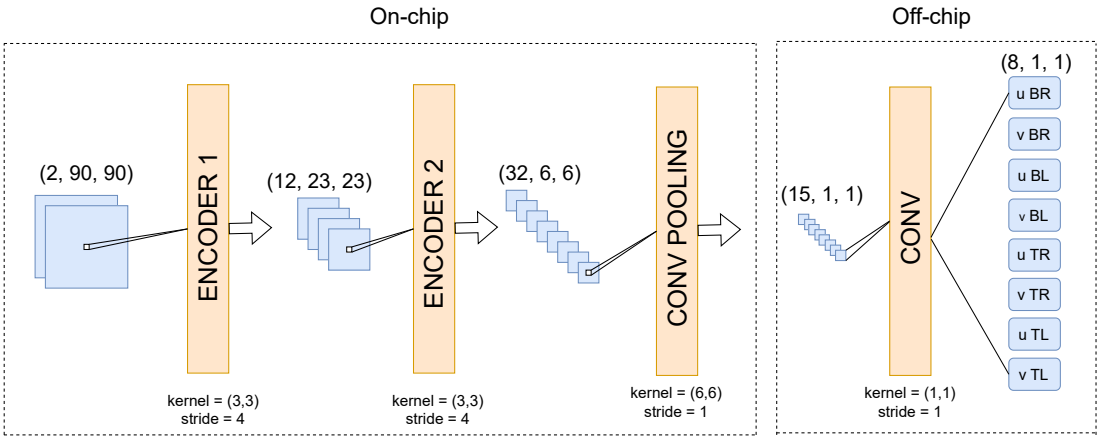


Fig. 15. Overview of the network implementation on the speck2e

on the speck2e the layer appears to be reaching a saturation point, after which it cannot produce more spikes. When the input spikes increase, the output spikes of the layer increase up to a certain point and then they start accumulating. When the inputs decrease again around 1000 ms, the accumulated outputs are released sequentially. Accumulation obviously results in information loss, because the timestamp of the spikes is delayed. Although it is still possible to retrieve the optical flow prediction by entering a predefined sequence (Figure 18), these settings are not ideal for real-time vision and control. If the network circuit received inputs from the DVS directly and the spikes between layers started to accumulate, this would result in a backlog, since the DVS would not be able to transmit more spikes to the network.

C. Weight Clamping

Although different methods to reduce the spikes were implemented in Section IV-F, the convolutional cores still cannot output as many spikes as necessary to predict the flow properly. This gap in performance between simulation and hardware is most likely due the weight values being larger than the thresholds.

To prove this, the network is tested with the same input sequence, but the weights are clamped to a maximum equal to the threshold and a minimum equal to the negative of the threshold. Figure 17 shows the effect of clamping on the weight distribution. In the first layer of the first encoder the quantized thresholds are approximately $\pm$ 30. Constraining the weights to the threshold values results in a reduction of information, because for every synaptic input, a neuron can spike at most once. This restricts the range of possible neuronal activity patterns, hence it reduces the accuracy of the network. However, the spike activity does not get stuck at the saturation point anymore (Figure 16 red line). With these settings, the information flows more smoothly throughout the network without accumulating.

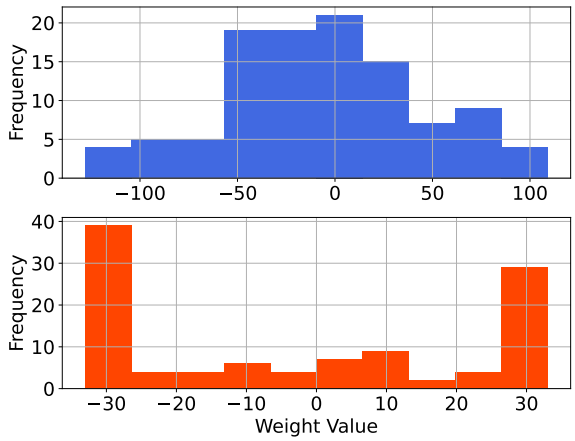


Fig. 17. Effect of clamping on weights distribution in first layer of first encoder.

D. Optical Flow Prediction

The spikes of the pooling layer are binned into frames and fed in the last off-chip layer. The resulting optical flow is shown in Figure 18.

IF-2-C on speck2e

When the weights are not clamped, the network prediction manages to follow the simulation result. However, especially at the peaks, the prediction from the speck2e deteriorates. This is because, for the network to be accurate at higher magnitudes of optical flow, more spikes need to be transmitted between the layers, which is not possible if the layers reach the saturation point.

IF-2-C on speck2e with clamped weights

By clamping the weights, the layers are able to send spikes without reaching the limit. However, the network parameters are way more different from the trained solution, hence the prediction loses accuracy. This proves that the limitations of the speck2e are encountered when using weights values with higher values than thresholds, as this results in a too high demand for spikes processing.

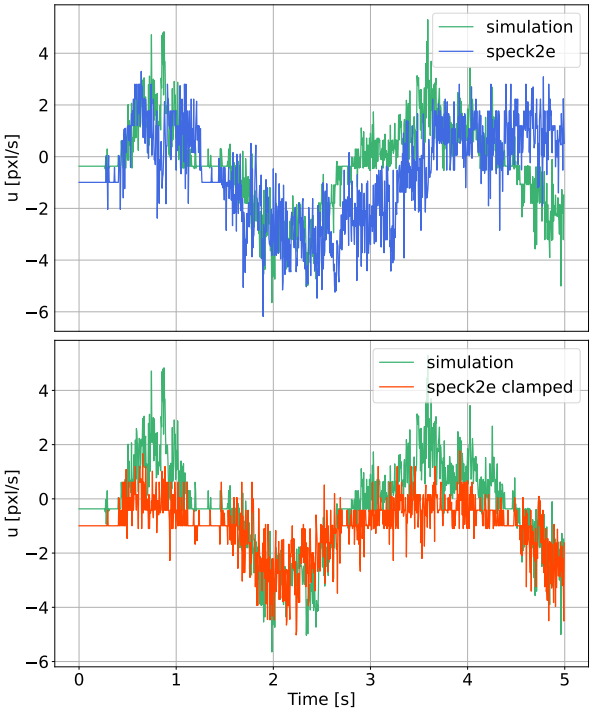


Fig. 18. Optical flow prediction for normal network and network with clamped weights on the speck2e.

It is interesting to notice that for negative flow values, the speck2e output is more accurate than for positive ones. This is likely because during training, the network associated fewer spikes with negative flow and more spikes with positive flow. To predict higher values of positive flow, the layers will need to spike more than allowed by the device limitations. However, for negative flow values, less spikes are required and a better flow estimate is obtained. Using clamped weights, the range of possible optical flow values is roughly [-4,1] pixel/s. To further improve the prediction, one could re-train the network by constraining the activity to obtain a range of possible optical flow that is symmetric in positive and negative direction.

VI. Results Discussion & Conclusion

In this paper, we presented a novel approach for optical flow determination using neuromorphic computing. The speck2e device from Synsense was used in the experiments and the network configuration was inspired by Paredes-Vallés et al. [18]. This device could result in lighter and more power efficient MAVs, as it includes sensing and computing into one board. It was shown that it is possible to estimate optical flow from event-based datasets in simulation with a RNN trained and converted to SNN. A mechanism of weight rescaling was applied on the network parameters to avoid information loss between the layers after the conversion. Although the simulation results seem promising, they are based on the assumption that neurons on board of the speck2e can have synaptic weight values higher than the thresholds and output a spike rate proportional to such weight. Uploading and testing a network with this characteristics is possible, however the resulting performance of the network on hardware is quite different than in simulation. It was noticed that the convolutional cores reach a limit on the number of synaptic operations, despite the regularization term used during training. This saturation point is reached if the weights are higher than the thresholds, since the potential reaches its limit much faster. The result is spikes accumulation which results a loss of information. It is possible to alter the parameters to make the network run without spikes accumulation. This is achieved by clamping the weight values to the threshold limits. Note that this severely reduces the range of possible weight values, hence it makes the network more limited and the prediction less exact.

This paper served to explore the limitations and challenges of training a speck2e-compatible IF network to estimate optical flow. With the help of this analysis it might be possible to further improve the performance of the network, by re-training it under the constraints of the device. A good improvement would be re-training the network to have its range of possible optical flow values to be symmetric in positive and negative direction. One limitation that is still constraining the speck2e is the limit on synaptic operations per second. This represent a complex challenge, considering that all the information has to be sent through the spikes between layers.

References

[1] Carlo Michaelis, Andrew B. Lehr, and Christian Tetzlaff. “Robust Trajectory Generation for Robotic Control on the Neuromorphic Research Chip Loihi”. In: *Frontiers in Neurorobotics* 14 (2020). ISSN: 1662-5218. DOI: 10.3389/fnbot.2020.589532. URL: https://www.frontiersin.org/articles/10.3389/fnbot.2020.589532.

[2] Raphaela Kreiser, Alpha Renner, and Yulia Sandamirskaya. “Error-driven learning for self-calibration in a neuromorphic path integration system”. In: Aug. 2019.

[3] Raphaela Kreiser et al. “Error estimation and correction in a spiking neural network for map formation in neuromorphic hardware”. In: May 2020, pp. 6134–6140. DOI: 10.1109/ICRA40945.2020.9197498.

[4] Catherine D Schuman et al. “A survey of neuromorphic computing and neural networks in hardware”. In: *arXiv preprint arXiv:1705.06963* (2017).

[5] Wolfgang Maass. “Networks of spiking neurons: The third generation of neural network models”. In: *Neural Networks* 10.9 (1997), pp. 1659–1671. ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(97)00011-7. URL: https://www.sciencedirect.com/science/article/pii/S0893608097000117.

[6] André Grüning and Sander M Bohte. “Spiking neural networks: Principles and challenges.” In: *ESANN*. Bruges. 2014.

[7] Caterina Caccavella et al. *Low-power event-based face detection with asynchronous neuromorphic hardware*. 2023. arXiv: 2312.14261 [cs.NE].

[8] David Drazen et al. “Toward real-time particle tracking using an event-based dynamic vision sensor”. In: *Experiments in Fluids* 51 (Nov. 2011), pp. 1465–1469. DOI: 10.1007/s00348-011-1207-y.

[9] Zhenjiang Ni et al. “Asynchronous event-based high speed vision for microparticle tracking”. In: *Journal of microscopy* 245 (Nov. 2011), pp. 236–44. DOI: 10.1111/j.1365-2818.2011.03565.x.

[10] Arnon Amir et al. “A Low Power, Fully Event-Based Gesture Recognition System”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[11] Enea Ceolini et al. “Hand-Gesture Recognition Based on EMG and Event-Based Camera Sensor Fusion: A Benchmark in Neuromorphic Computing”. In: *Frontiers in Neuroscience* 14 (2020). ISSN: 1662-453X. DOI: 10.3389/fnins.2020.00637. URL: https://www.frontiersin.org/articles/10.3389/fnins.2020.00637.

[12] Haiyang Chao, Yu Gu, and Marcello Napolitano. “A Survey of Optical Flow Techniques for Robotics Navigation Applications”. In: *Journal of Intelligent I& Robotic Systems* 73 (May 2013). DOI: 10.1007/s10846-013-9923-6.

[13] S. S. Beauchemin and J. L. Barron. “The Computation of Optical Flow”. In: 27.3 (Sept. 1995), pp. 433–466. ISSN: 0360-0300. DOI: 10.1145/212094.212141. URL: https://doi.org/10.1145/212094.212141.

[14] Javaan Chahl, Mandyam Srinivasan, and Shaowu Zhang. “Landing Strategies in Honeybees and Applications to Uninhabited Airborne Vehicles”. In: *I. J. Robotic Res.* 23 (Feb. 2004), pp. 101–110. DOI: 10.1177/0278364904041320.

[15] Harald E. Esch and John E. Burns. “Distance Estimation by Foraging Honeybees”. In: *Journal of Experimental Biology* 199.1 (Jan. 1996), pp. 155–162. ISSN: 0022-0949. DOI: 10.1242/jeb.199.1.155. eprint: https://journals.biologists.com/jeb/article-pdf/199/1/155/3107314/jexbio_199_1_155.pdf. URL: https://doi.org/10.1242/jeb.199.1.155.

[16] Mandyam Srinivasan. “Honeybees as a Model for the Study of Visually Guided Flight, Navigation, and Biologically Inspired Robotics”. In: *Physiological reviews* 91 (Apr. 2011), pp. 413–60. DOI: 10.1152/physrev.00005.2010.

[17] Guillermo Gallego et al. “Event-based vision: A survey”. In: *IEEE transactions on pattern analysis and machine intelligence* 44.1 (2020), pp. 154–180.

[18] Federico Paredes-Vallés et al. “Fully neuromorphic vision and control for autonomous drone flight”. In: *arXiv preprint arXiv:2303.08778* (2023).

[19] *Loihi 2: A New Generation of Neuromorphic Computing*. URL: https://www.intel.com/content/www/us/en/research/neuromorphic-computing.html.

[20] *Speck: Event-Driven Neuromorphic SoC — Synsense*. URL: https://www.synsense.ai/products/speck-2/.

[21] *Neuromorphic Intelligence I& Application Solutions — Synsense*. URL: https://www.synsense.ai/.

[22] Ole Richter et al. *Speck: A Smart event-based Vision Sensor with a low latency 327K Neuron Convolutional Neuronal Network Processing Pipeline*. 2023. arXiv: 2304.06793 [cs.NE].

[23] Fernando Perez-Peña, Alejandro Linares-Barranco, and Elisabetta Chicca. “An approach to motor control for spike-based neuromorphic robotics”. In: *2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) Proceedings*. 2014. DOI: 10.1109/BioCAS.2014.6981779.

[24] J.C. Gallacher and J.M. Fiore. “Continuous time recurrent neural networks: a paradigm for evolvable analog controller circuits”. In: *Proceedings of the IEEE 2000 National Aerospace and Electronics Conference. NAECON 2000. Engineering Tomorrow (Cat. No.00CH37093)*. 2000, pp. 299–304. DOI: 10.1109/NAECON.2000.894924.

[25] D. Roggen et al. “Hardware spiking neural network with run-time reconfigurable connectivity in an autonomous robot”. In: *NASA/DoD Conference on Evolvable Hardware, 2003. Proceedings.* 2003, pp. 189–198. DOI: 10.1109/EH.2003.1217666.

[26] Di Hu et al. “Digital implementation of a spiking neural network (SNN) capable of spike-timing-dependent plasticity (STDP) learning”. In: *14th IEEE International Conference on Nanotechnology*. 2014, pp. 873–876. DOI: 10.1109/NANO.2014.6968000.

[27] Patrick Rocke et al. “Reconfigurable Hardware Evolution Platform for a Spiking Neural Network Robotics Controller”. In: *Reconfigurable Computing: Architectures, Tools and Applications*. Ed. by Pedro C. Diniz et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 373–378. ISBN: 978-3-540-71431-6.

[28] Giacomo Indiveri and Paul Verschure. “Autonomous vehicle guidance using analog VLSI neuromorphic sensors”. In: *Artificial Neural Networks — ICANN’97*. Ed. by Wulfram Gerstner et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 811–816. ISBN: 978-3-540-69620-9.

[29] Scott Koziol, Stephen Brink, and Jennifer Hasler. “A Neuromorphic Approach to Path Planning Using a Reconfigurable Neuron Array IC”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 22.12 (2014), pp. 2724–2737. DOI: 10.1109/TVLSI.2013.2297056.

[30] Jesse J. Hagenaars et al. “Evolved Neuromorphic Control for High Speed Divergence-based Landings of MAVs”. In: *CoRR* abs/2003.03118 (2020). arXiv: 2003.03118. URL: https://arxiv.org/abs/2003.03118.

[31] Steven Abreu et al. “Flow Cytometry With Event-Based Vision and Spiking Neuromorphic Hardware”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2023, pp. 4139–4147.

[32] Kangrui Du et al. *Temporal Flexibility in Spiking Neural Networks: A Novel Training Method for Enhanced Generalization Across Time Steps*. 2024. URL: https://openreview.net/forum?id=RmQAKu1wCe.

[33] *Sinabs (Sinabs Is Not A Brain Simulator)*. URL: https://sinabs.readthedocs.io/en/1.2.8/index.html.

[34] *Samna*. URL: https://pypi.org/project/samna/.

[35] Yongqiang Cao, Yang Chen, and Deepak Khosla. "Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition". In: *International Journal of Computer Vision* 113 (May 2015), pp. 54–66. DOI: 10.1007/s11263-014-0788-3.

[36] Bodo Rueckauer et al. "Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification". In: *Frontiers in Neuroscience* 11 (2017). ISSN: 1662-453X. DOI: 10.3389/fnins.2017.00682. URL: https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2017.00682.

[37] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. "A Unifying Contrast Maximization Framework for Event Cameras, with Applications to Motion, Depth, and Optical Flow Estimation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3867–3876. DOI: 10.1109/CVPR.2018.00407.

[38] Alex Zihao Zhu et al. "Unsupervised Event-based Learning of Optical Flow, Depth, and Egomotion". In: *CoRR* abs/1812.08156 (2018). arXiv: 1812.08156. URL: http://arxiv.org/abs/1812.08156.

[39] Federico Paredes-Vallés, Jesse J. Hagenaars, and Guido de Croon. "Self-Supervised Learning of Event-Based Optical Flow with Spiking Neural Networks". In: *CoRR* abs/2106.01862 (2021). arXiv: 2106.01862. URL: https://arxiv.org/abs/2106.01862.

[40] Elias Mueggler et al. "The Event-Camera Dataset and Simulator: Event-based Data for Pose Estimation, Visual Odometry, and SLAM". In: *CoRR* abs/1610.08336 (2016). arXiv: 1610.08336. URL: http://arxiv.org/abs/1610.08336.

APPENDIX A
OPTICAL FLOW COLORMAP

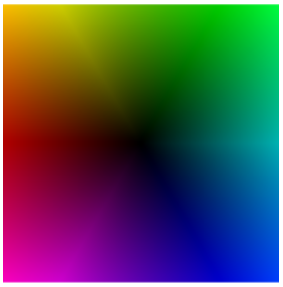


Fig. 19. Optical flow color map.

APPENDIX B
NETWORK CONFIGURATION DETAILS

This section includes the network configuration details, meaning the channels number, size, kernel, stride and padding for each convolutional layer.

TABLE II
NETWORK ARCHITECTURE FOR RNN-2-S ON CYBERZOO DATASET.

	Layer	In Channels	Out Channels	In Size	Out Size	Kernel	Stride	Padding
Encoder 0	Fwd.	2	4	180	90	3	2	1
	Rec.	4	4	90	90	3	1	0
Encoder 1	Fwd.	4	8	90	23	3	4	1
	Rec.	8	8	23	23	3	1	0
	Pooling	8	8	23	1	23	1	0
	Prediction	8	8	1	1	1	1	0

TABLE III
NETWORK ARCHITECTURE FOR RNN-2-C ON CYBERZOO DATASET.

	Layer	In Channels	Out Channels	In Size	Out Size	Kernel	Stride	Padding
Encoder 0	Fwd. 1	2	4	180	90	3	2	1
	Fwd. 2	8	8	90	90	3	1	1
	Rec.	8	4	90	90	3	1	0
Encoder 1	Fwd. 1	8	16	90	23	3	4	1
	Fwd. 2	32	32	23	23	3	1	1
	Rec.	32	16	23	23	3	1	0
	Pooling	32	8	23	1	23	1	0
	Prediction	8	8	1	1	1	1	0

TABLE IV
NETWORK ARCHITECTURE FOR RNN-2-C ON SPECK.

	Layer	In Channels	Out Channels	In Size	Out Size	Kernel	Stride	Padding
Encoder 0	Fwd. 1	2	6	90	23	3	4	1
	Fwd. 2	12	12	23	23	3	1	1
	Rec.	12	6	23	23	3	1	0
Encoder 1	Fwd. 1	12	16	23	6	3	4	1
	Fwd. 2	32	32	6	6	3	1	1
	Rec.	32	16	6	6	3	1	0
	Pooling	32	15	6	1	6	1	0
	Prediction	15	8	1	1	1	1	0

APPENDIX C
EFFECT OF HYPERPARAMETERS ON SYNOPS

This section includes the plots of number of synaptic operations and maximum number of operations per neuron over time, for the first convolutional layer for different network configurations. In each pair of plots, a single hyperparameter was altered to observe the effect on the synaptic operations.

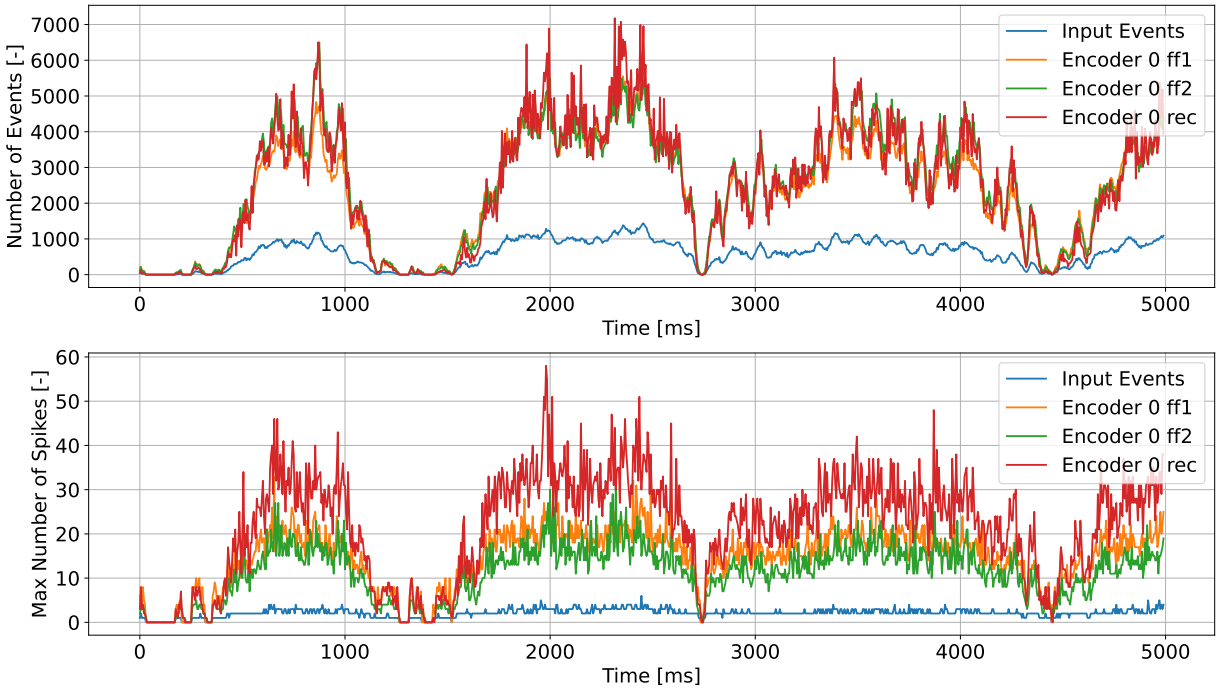


Fig. 20. Spike activity in RNN-2-C with stride 2 in first encoder.

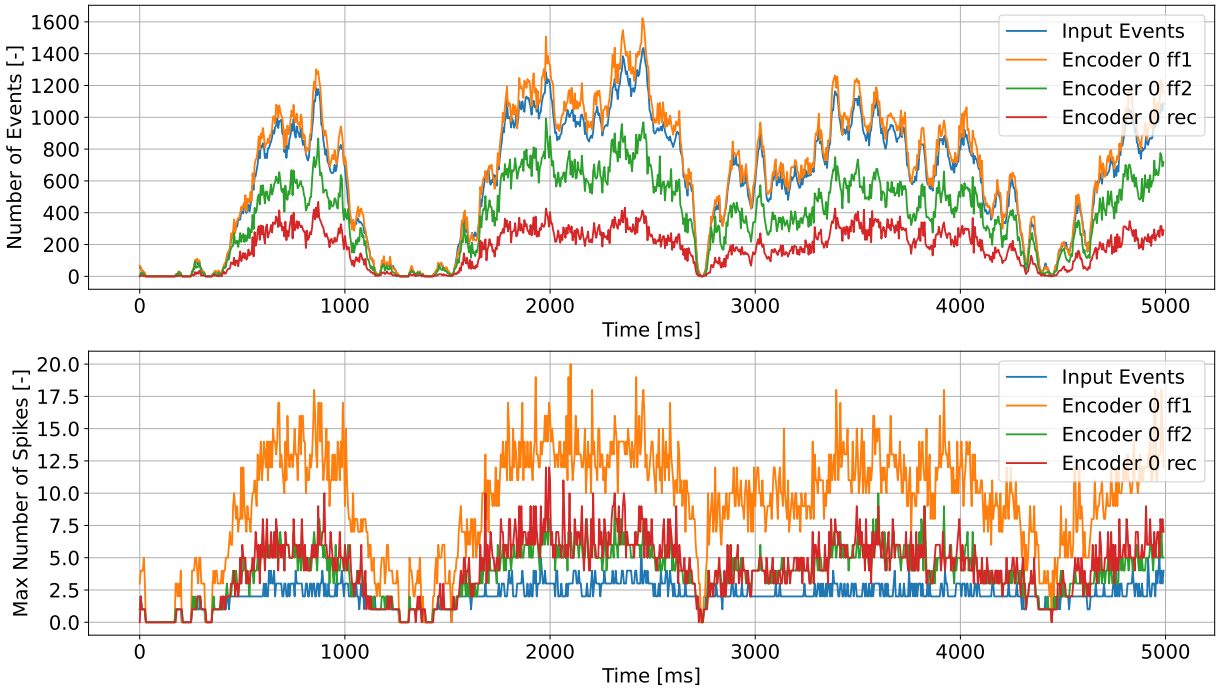


Fig. 21. Spike activity RNN-2-C with stride 4 in first encoder.

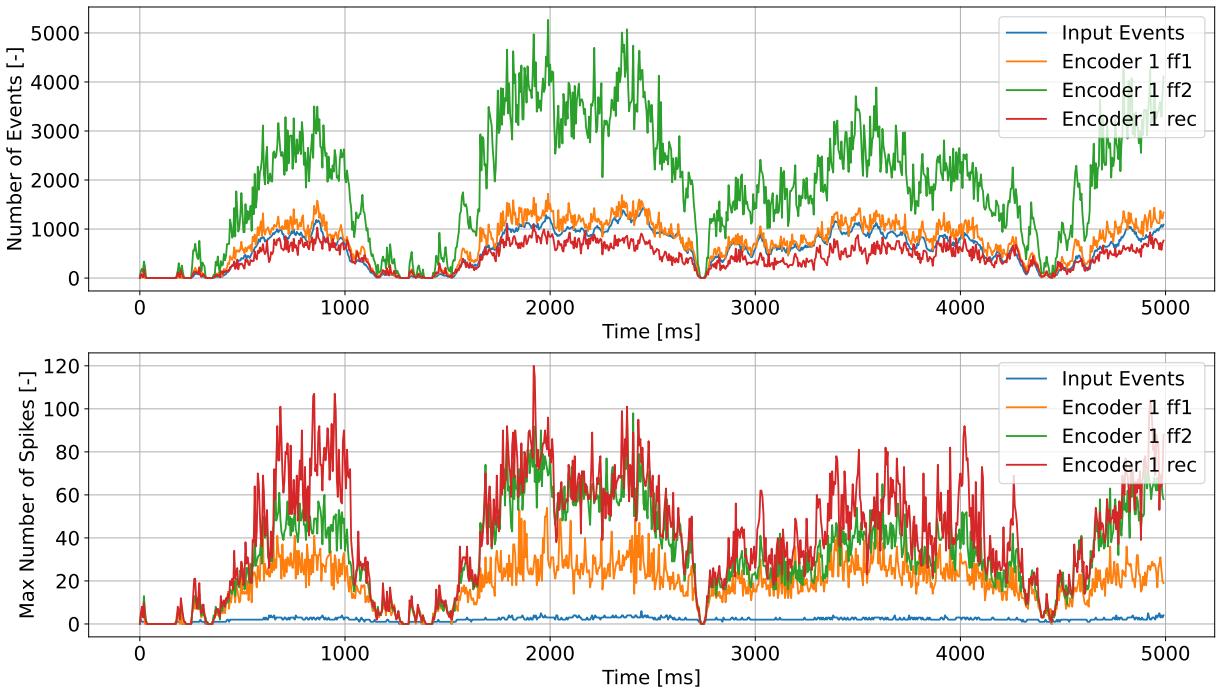


Fig. 22. RNN-2-B with increased number of channels.

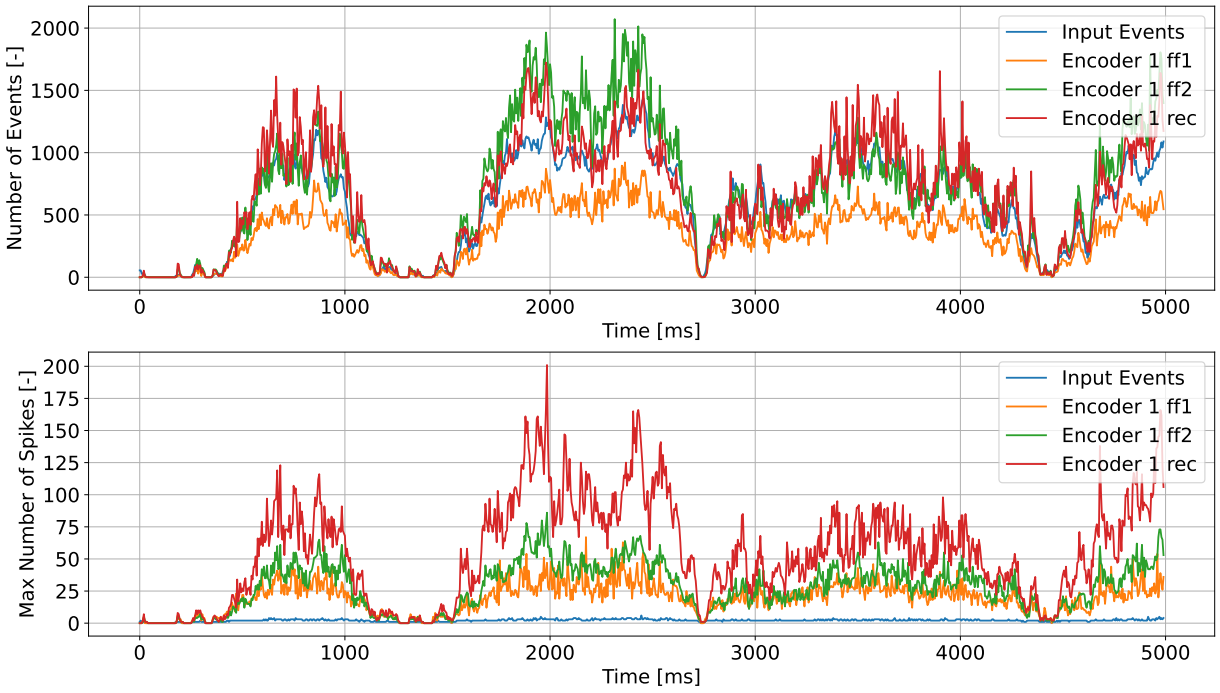
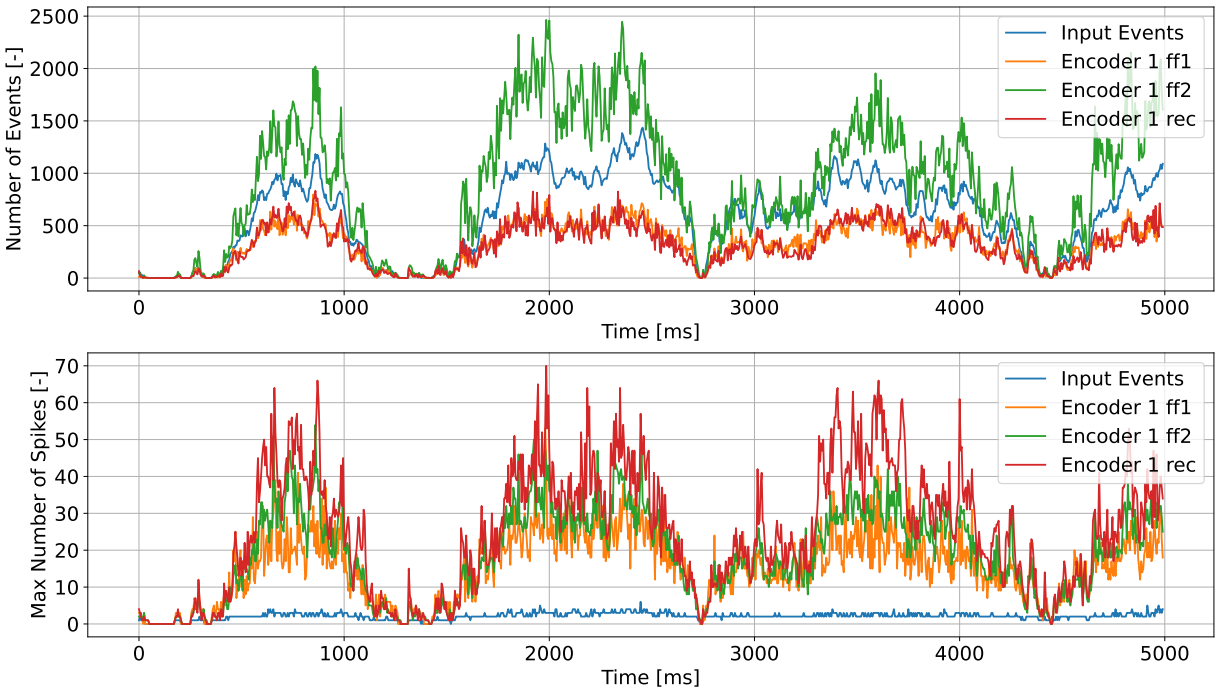


Fig. 23. RNN-2-C with decreased number of channels.



H

Fig. 24. RNN-2-C after 80 epochs.

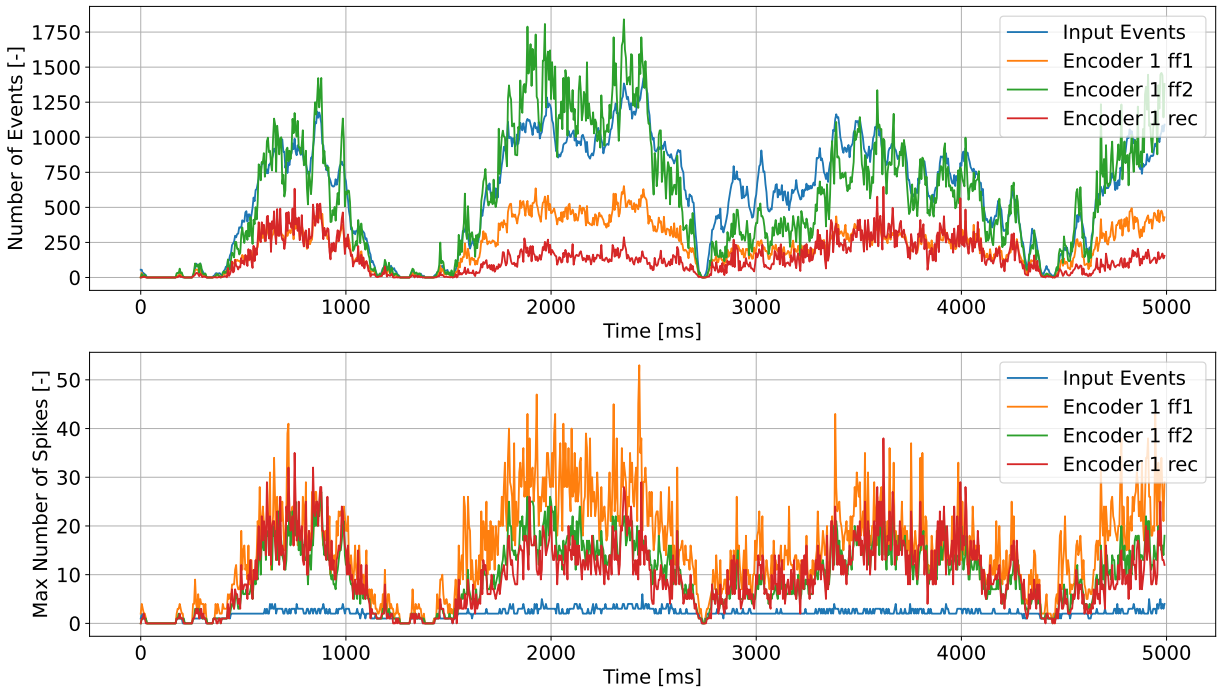


Fig. 25. RNN-2-C after 10 epochs.