2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Adaptive-SpikeNet: Event-based Optical Flow Estimation using Spiking Neural Networks with Learnable Neuronal Dynamics

Adarsh Kumar Kosta and Kaushik Roy

***Abstract*— Event-based cameras have recently shown great potential for high-speed motion estimation owing to their ability to capture temporally rich information asynchronously. Spiking Neural Networks (SNNs), with their neuro-inspired event-driven processing can efficiently handle such asynchronous data, while neuron models such as the leaky-integrate and fire (LIF) can keep track of the quintessential timing information contained in the inputs. SNNs achieve this by maintaining a dynamic state in the neuron memory, retaining important information while forgetting redundant data over time. Thus, we posit that SNNs would allow for better performance on sequential regression tasks compared to similarly sized Analog Neural Networks (ANNs). However, deep SNNs are difficult to train due to vanishing spikes at later layers. To that effect, we propose an adaptive fully-spiking framework with learnable neuronal dynamics to alleviate the spike vanishing problem. We utilize surrogate gradient-based backpropagation through time (BPTT) to train our deep SNNs from scratch. We validate our approach for the task of optical flow estimation on the Multi-Vehicle Stereo Event-Camera (MVSEC) dataset and the DSEC-Flow dataset. Our experiments on these datasets show an average reduction of $\sim 13\%$ in average endpoint error (AEE) compared to state-of-the-art ANNs. We also explore several down-scaled models and observe that our SNN models consistently outperform similarly sized ANNs offering $\sim 10\%$-16% lower AEE. These results demonstrate the importance of SNNs for smaller models and their suitability at the edge. In terms of efficiency, our SNNs offer substantial savings in network parameters ($\sim 48.3\times$) and computational energy ($\sim 10.2\times$) while attaining $\sim 10\%$ lower EPE compared to the state-of-the-art ANN implementations.**

I. INTRODUCTION

Research in the fields of neuroscience, machine learning and robotics has been highly inspired by the behaviour of biological species in their natural environments. Be it a house-fly seamlessly navigating through cluttered spaces, an osprey diving to catch a fish, a squirrel jumping from one branch to the other or a bee precisely landing on a swaying flower [1]–[4]. All these behaviours require a comprehensive understanding of the environment dynamics at an extremely low-latency and energy. Visualizing the motion field in such environments, often referred to as optical flow is one of the fundamental tasks involved. The ability to perform accurate and low-latency optical estimations would greatly benefit modern day artificial intelligence (AI)-powered robots such as drones, ground robots, and autonomous vehicles. It will further fuel more complex tasks such as obstacle detection/avoidance and path planning, enabling seamless autonomous navigation. Modern AI aims to achieve this using data from sensors such as frame-based cameras coupled with Analog Neural Networks (ANNs[1]). However, such models when optimized for performance turn out to be compute and memory intensive while having a high-latency. On the other hand, smaller models suitable for the edge suffer in terms of performance and require constant communication with the cloud. In light of this, there is a need to deliberate over our outlook towards tackling such tasks and rethink the processing pipeline all the way from sensors, to algorithms, to underlying hardware.

For sequential regression tasks with rapidly changing inputs, the timing information between the inputs is crucial and needs to be captured at a high temporal resolution. Frame-based cameras fail in such scenarios due to their fixed and low temporal resolution. They also suffer from motion blur, temporal aliasing and poor image quality in low light and high dynamic range scenes. Event-based cameras, such as the Dynamic Vision Sensor (DVS) [5], [6], overcome these issues by asynchronously sampling the log intensity changes at every pixel independently. They promise a high temporal resolution in the order of microseconds, a high dynamic range (140dB vs 60dB) and extremely low power consumption (10mW vs 3W) compared to frame-based cameras. However, ANN-based methods which were originally designed for frame-based images turn out to be incompatible at directly handling the binary and asynchronous outputs generated by event cameras. ANNs discard the temporal ordering of inputs by representing them as channels and perform sub-optimal stateless computations.

Spiking Neural Networks (SNNs), often referred to as the third generation of neural networks, offer asynchronous event-driven compute that fits naturally with event data. SNNs perform computations only at the arrival of inputs generating progressively sparser outputs as the network depth increases. Due to the implicit recurrence, SNNs can preserve the input timing information in the neuronal state called 'membrane potential' based on parameters such as neruon firing threshold and leak. This makes them inherently suitable for handling sequential tasks efficiently. SNNs implemented on neuromorphic processors such as Loihi [7] and TrueNorth [8] can offer several orders of magnitude lower energy consumption and latency compared to traditional von Neumann architectures. Over the past few years, direct SNN training for event-based vision has been limited to small scale and discrete problems such as image classification and action recognition on simplistic datasets like N-MNIST [9],

All authors are with Purdue University, West Lafayette, IN 47907, USA
{akosta, kaushik}@purdue.edu

[1]We refer to deep learning networks as ANNs due to their analog nature of inputs/computations, even though the underlying computing hardware can be digital.

DVS128 Gesture [10] etc. This is due to the fact that deep SNNs are difficult to train, mainly owing to vanishing spikes at deeper layers.

To that effect, we propose an adaptive fully-spiking framework with learnable neuronal dynamics for the complex regression task of optical flow estimation. Our proposed framework overcomes the spike vanishing problem in SNNs, effectively capturing the temporal information contained in the inputs and allowing training of deep SNNs. Our main contributions are as follows:

leftmargin=*

- We propose a fully-spiking framework with adaptive leaky-integrate and fire (LIF) neurons that can be trained from scratch using surrogate gradients during backpropagation, for the sequential regression task of event-based optical flow estimation.
- We show that our fully-spiking models outperform state-of-the-art ANN [11] and hybrid SNN-ANN [12] models, both in terms of performance and compute efficiency on MVSEC [13] and DSEC-Flow [14], [15] datasets.
- We demonstrate the importance of effectively capturing temporal information by analyzing various model sizes and observe that the performance difference between correspondingly sized SNNs and ANNs remains consistent upon reducing the model size.

II. Related Works

With event cameras showcasing great potential for low-latency optical flow estimation, there have been several (optimization and non-learning based) works in the recent years exploring this research area [16]–[18]. In learning-based ANN methods, the first ANN employed self-supervised learning (SSL) for event-based flow estimation and was proposed in [11] inspired by the U-Net [19] architecture. Another similar approach was presented in [20] which predicted optical flow using depth and pose estimates. These approaches used separate channels to provide encoded timing information as inputs to the ANN model which was sub-optimal. Researchers in [21], introduced a voxel-based event-input representation inspired by [16] and a contrast maximization loss [18] removing the dependency on grayscale images for loss computations. Authors in [22] extend this to image reconstruction with events and propose a new lightweight architecture called Fire-FlowNet for flow estimation. In contrast to the above SSL pipelines, authors in [23] utilized synthetic data from an event simulator [24] to train the networks proposed in [11], [21] and obtained significantly better performance. However, all these works do not specifically focus on exploiting the timing information or temporal ordering between inputs. A recent supervised work called E-RAFT [15] explored the importance of timing information by explicitly using temporal history and leveraging correlation volumes for iterative flow refinement.

Regarding learning-based SNN works for optical flow estimation, an STDP (Spike Timing Dependent Plasticity) based convolutional SNN was presented in [25] but was limited to relatively simplistic flow behaviour. Following this, authors in [12] presented a hybrid SNN-ANN version of [11] with a SNN encoder and ANN decoder, outperforming it and establishing the utility of SNNs. Researchers in [26] expanded this framework by performing sensor fusion and adding a secondary ANN-based encoder for frame-based data. In purely SNN-based approaches, researchers in [27] explored parameter initialization, surrogate gradients and adaptive neuronal mechanisms for flow estimation. However, their results were inconclusive in several scenarios for the explored neuronal mechanisms. Although, we agree that adding explicit recurrence improves performance as pointed out in [15], [27], we believe that relaxing the constraints on the adaptivity of neuronal dynamics and exploiting the implicit recurrence offered by SNNs can lead to better performance without additional parameters and learning overheads. Works such as [28]–[30] show the importance of neuronal dynamics towards improving performance, however, they target more straightforward classification tasks. We explore their extendability to a complex regression task of optical flow estimation through this work.

III. Method

A. Sensors and Input representation

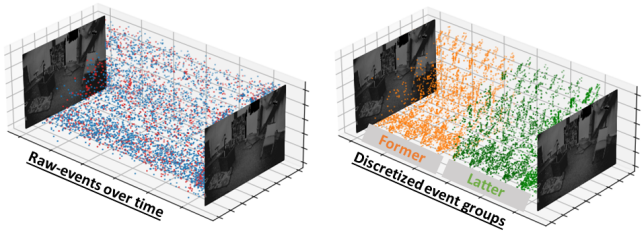

Fig. 1: (*left*) Raw event stream between two consecutive frames. (*right*) Event bins in former and latter event groups.

Event-based cameras, inspired by the biological retina [31], sample the log intensity changes at each discrete pixel asynchronously and independently. Any change in the log intensity (I) over a specified threshold (θ) is recorded as a discrete event at that pixel location, (i.e., $\|\log(I_{t+1}) - \log(I_t)\| \geq \theta$). The data is generated in the Address Event Representation (AER) format comprising a tuple $\{x, y, t, p\}$, with (x, y) representing pixel locations, (t) representing the camera timestamp and (p) representing the (ON/OFF) polarity of the intensity change.

In this work, we utilize the discretized event volume representation as presented in [21]. For a set of N input events $\{(x_i, y_i, t_i, p_i)\}_{i\in[1,N]}$ between two consecutive grayscale images and a set of B event bins to be created within this event volume, the discretized event volume is generated using bilinear sampling as follows:

$$t_i^* = (B-1)(t_i - t_1)/(t_N - t_1) \tag{1}$$

$$V(x,y,t) = \sum_i p_i k_b(x - x_i) k_b(y - y_i) k_b(t - t_i^*) \tag{2}$$

$$k_b = max[0, 1 - |a|] \tag{3}$$

where $k_b(a)$ is the bilinear sampling kernel from [32]. We further process the obtained discretized voxel into former and

latter groups of event bins corresponding to each (ON/OFF) polarity leading to a 4-channeled representation involving former and latter event groups, as described in [12]. These groups facilitate flow estimation between equidistant bins (5) to obtain as many flow predictions as possible in the event volume while avoiding different temporal scales. Each channel thus contains a total of $T = B/2$ event frames that are passed sequentially as timesteps through the network. The idea behind passing a multi-channel representation at each timestep is to allow the network to learn a larger temporal correlation directly while capturing the short term temporal correlation in neuronal dynamics over timesteps. This representation preserves the spatio-temporal information in the event stream while offering high energy-efficiency due to controlled number of timesteps (T) for sequential processing. Fig. 1 shows the former and latter event groups based event input representation.

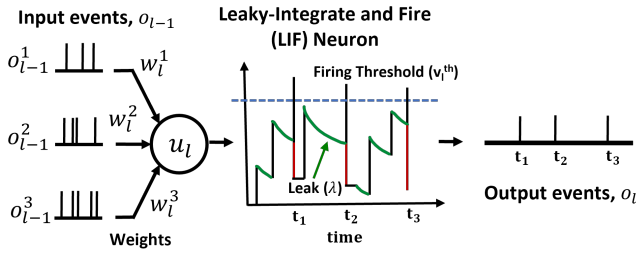

Fig. 2: Leaky-Integrate and Fire (LIF) neuron. The firing threshold (v_l^{th}) and leak factor (λ) are learnable parameters.

B. Neuron Model

There exist several biologically inspired neuron models of which the Leaky-Integrate-and-Fire (LIF) [33] is the most widely used due to its immense capability of storing and recalling information, yet being simple enough to not explode the model parameters. The LIF neuron allows to accumulate information over time into the neuronal state called 'membrane potential' (u), while also enabling controlled forgetting over time (through leak, λ). The LIF neuron model in an SNN can be described as:

$$\boldsymbol{u}_l^t = \lambda_l \boldsymbol{u}_l^{t-1} + \boldsymbol{W}_l o_{l-1}^t - v_l^{th} \boldsymbol{o}_l^{t-1} \tag{4}$$

where $\boldsymbol{u}_l^t$ represents the membrane potential of the layer l at timestep t, $\boldsymbol{W}_l$ represents the synaptic weights connecting layers $l-1$ and l, o_l represents the binary output spike at layer l, v_l^{th} is the neuronal firing threshold and λ_l represents the neuronal leak factor for layer l. The first term in Eq.4 denotes the leakage in membrane potential, the second term computes the summation of weighted output spikes from the previous layer, and the third term denotes the reduction of membrane potential upon generation of an output spike at the current layer. This reduction by an amount equal to the firing threshold (v_i^{th}) is termed as 'soft reset', while a reset to zero is termed as 'hard reset'. The generation of output spikes follows the following equation at each timestep:

$$\boldsymbol{z}_l^t = \boldsymbol{u}_l^t / v_l^{th} - 1, \quad \boldsymbol{o}_l^t = \begin{cases} 1, & \text{if } \boldsymbol{z}_l^t > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

These operations are carried out over all timesteps in each neuron enabling event-driven computations. The threshold governs the average integration time of inputs while the leak controls the amount of membrane potential retained across timesteps. Traditionally, the neuronal firing threshold (v^{th}) and the leak factor (λ) are fixed *a priori*. We posit that making v^{th} and λ parameters trainable would allow the network to maintain sufficient spiking activity eliminating vanishing spikes and remember temporal information using the membrane potential, allowing deep SNNs to learn complex tasks [28]. Fig 2 shows the LIF neuron dynamics.

C. Network Architecture

We explore two types of fully-spiking architectures: an encoder-decoder based multi-scale architecture based on U-net [19] and a lightweight single-scale architecture called Fire-FlowNet [22]. We evaluate several network sizes for the first architecture by subsequently scaling down the number of channels by a factor of 2 at each layer. The Fire-FlowNet [22] architecture performs no output downsampling and thus the input shape is maintained throughout the network.

A forward pass involves passing the 4-channeled input representation (former and latter groups) sequentially over T timesteps. At each timestep, the membrane potentials are updated (Eq. 4) and output spikes are generated (Eq. 5). The binary activations at every layer in the first type of architecture are downsampled/upsampled at each timestep before being forwarded to the next layer. The decoder layers upsample the provided input and also produce intermediate multi-scale flow predictions for each timestep. The flow prediction layers accumulate the flow over all timesteps to generate the final *Tanh* activated full-scale flow prediction. Fire-FlowNet involves a similar forward pass but lacks any upsampling/downsampling layers. These architectures and discussed operations are illustrated in Fig. 3.

D. Self-supervised Loss

The Multi-Vehicle Stereo Event Camera Dataset (MVSEC) [13] used in this work lacks reliable ground-truth labels. Thus, we adopt a self-supervised approach for training on this dataset [34]. The overall loss function is:

$$L^{\mathrm{u}} = l_{\mathrm{photo}} + \alpha l_{\mathrm{smooth}} \tag{6}$$

where l_{photo} and l_{smooth} represent photometric and, smoothness loss, respectively, and α denotes the weighting factor for smoothness loss.

1) Photometric Loss: The photometric loss imposes the brightness consistency assumption – the intensity moving from pixel location (x,y) at time t to pixel location $(x+dx, y+dy)$ at time $(t+dt)$ remains the same. It is computed using two consecutive grayscale images ($I_t(x,y)$, $I_{t+dt}(x,y)$) and the predicted optical flow (u,v). A spatial transformer [32] is used to inversely warp the second grayscale image ($I_{t+dt}(x,y)$) using the predicted optical flow (u,v). The photometric loss then minimizes the difference between the first grayscale image and the warped image as follows:

$$l_{photo} = \sum_{x,y} \rho(I_t(x,y) - I_{t+dt}(x+u, y+v)) \tag{7}$$

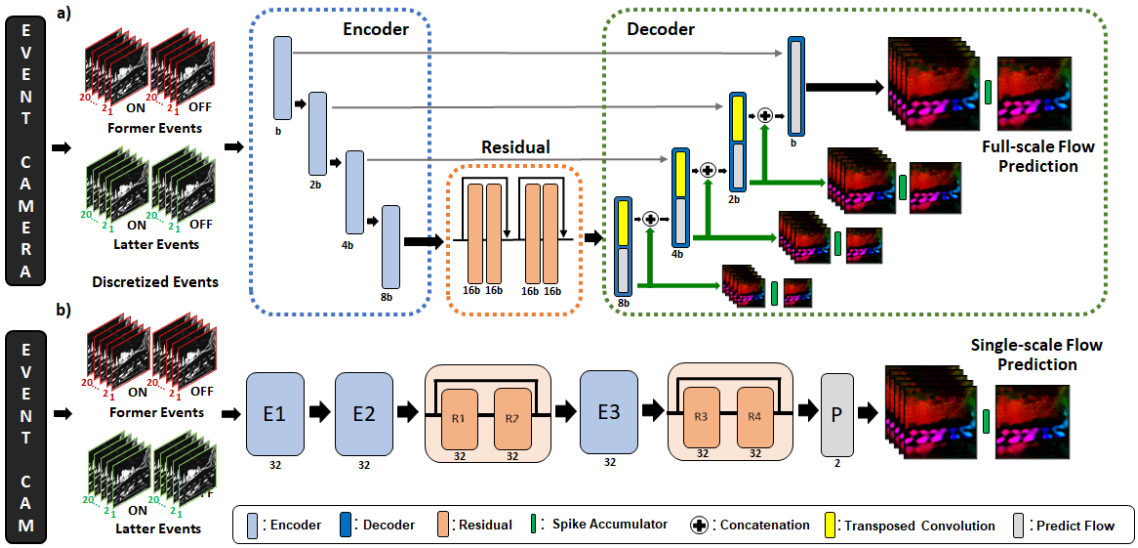

Fig. 3: Fully-spiking architectures based on a) U-net [19] and b) Fire-FlowNet [15]. Best viewed in color.

where ρ is the robust Charbonnier loss $\rho(x) = (x^2 + \eta^2)^r$ used for outlier rejection [35]. We set r=0.45 and η=1e^{-3}.

2) Smoothness Loss: Smoothness loss reduces the erratic variations in optical flow predictions at the edges by adding a regularization as follows:

$$l_{smooth} = \sum_j \sum_i (\|u_{i,j} - u_{i+1,j}\| + \|u_{i,j} - u_{i,j+1}\| + \|v_{i,j} - v_{i+1,j}\| + \|v_{i,j} - v_{i,j+1}\|) \quad (8)$$

where $u_{i,j}$ and $v_{i,j}$ are the flow estimates at pixel location (i, j) in x and y directions, respectively.

E. Supervised Loss

We also evaluate our proposed architectures on the DSEC-Flow dataset [14], [15] which has much better ground-truth flow maps compared to the MVSEC dataset. Hence, we perform supervised learning on this dataset. Given the predicted optical flow (u_{pred}, v_{pred}) and the ground-truth optical flow (u_{gt}, v_{gt}), the total loss L^s is computed as the mean squared error (MSE) as follows:

$$L^s = \frac{1}{n}\sum_{i=1}^{K}(u_{pred} - u_{gt})^2 + \frac{1}{n}\sum_{i=1}^{K}(v_{pred} - v_{gt})^2 \quad (9)$$

where K is the number of pixels with non-zero flow.

F. Surrogate gradient and Backpropagation-through time

Once the final loss (L) (in both self-supervised and supervised cases) is obtained, the next step is to perform backpropagation and compute the loss gradients with respect to the network parameters. However, unlike ANNs, gradient computation in SNNs is not straightforward.

The spike generation mechanism of an LIF neuron results in a hard threshold function which is non-differentiable. Thus, there is a need to approximate its gradient using a surrogate function. We use the inverse tangent function [29] as an approximation to the gradient of the LIF neuron, since it is widely used and is computationally inexpensive.

$$\frac{\partial o_l^t}{\partial z_l^t} = \begin{cases} \frac{1}{1+\gamma {z_l^t}^2}, & \text{if } 1 - |z_l^t| > 0 \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

$$\frac{\partial o_l^t}{\partial u_l^t} = \frac{\partial o_l^t}{\partial z_l^t}\frac{\partial z_l^t}{\partial u_l^t} = \frac{\partial o_l^t}{\partial z_l^t}\frac{1}{v_l^{th}} \quad (11)$$

The network also needs to be unrolled in time for all timesteps and the errors ($\frac{\partial L}{\partial o_l}$) need to be backpropagated using the surrogate gradient and Backpropagation Through Time (BPTT) [36]. Once all the gradients are obtained, the weight update can computed as the sum of gradients over each time-step. The weight updates for the SNN layer l are described as follows:

$$\triangle \boldsymbol{W}_l = \sum_t \frac{\partial \mathrm{L}}{\partial o_l^t}\frac{\partial o_l^t}{\partial z_l^t}\frac{\partial z_l^t}{\partial u_l^t}\frac{\partial u_l^t}{\partial w_l} = \sum_t \frac{\partial \mathrm{L}}{\partial o_l^t}\frac{\partial o_l^t}{\partial z_l^t}\frac{1}{v_l^{th}}o_{l-1}^t \quad (12)$$

Similarly, the threshold and leak updates are given by:

$$\triangle v_l^{th} = \sum_t \frac{\partial \mathrm{L}}{\partial o_l^t}\frac{\partial o_l^t}{\partial z_l^t}\frac{\partial z_l^t}{\partial v_l^{th}} = \sum_t \frac{\partial \mathrm{L}}{\partial o_l^t}\frac{\partial o_l^t}{\partial z_l^t}\left(\frac{-v_l^{th}o_l^{t-1} - u_l^t}{(v_l)^2}\right) \quad (13)$$

$$\triangle \lambda_l = \sum_t \frac{\partial \mathrm{L}}{\partial o_l^t}\frac{\partial o_l^t}{\partial u_l^t}\frac{\partial u_l^t}{\partial \lambda_l} = \sum_t \frac{\partial \mathrm{L}}{\partial o_l^t}\frac{\partial o_l^t}{\partial u_l^t}u_l^{t-1} \quad (14)$$

We use layer-wise learnable thresholds and leaks which lead to a negligible increase in the model size. Using per channel thresholds and leaks does not lead to a significant performance improvement as highlighted in [28].

IV. Experiments

A. Datasets and Training Details

1) MVSEC: The MVSEC [13] dataset contains four indoor flying, two outdoor day driving and, three outdoor night driving sequences. We perform training in SSL fashion using loss functions discussed in Section. III-D on the *outdoor_day*2 driving sequence and evaluate on

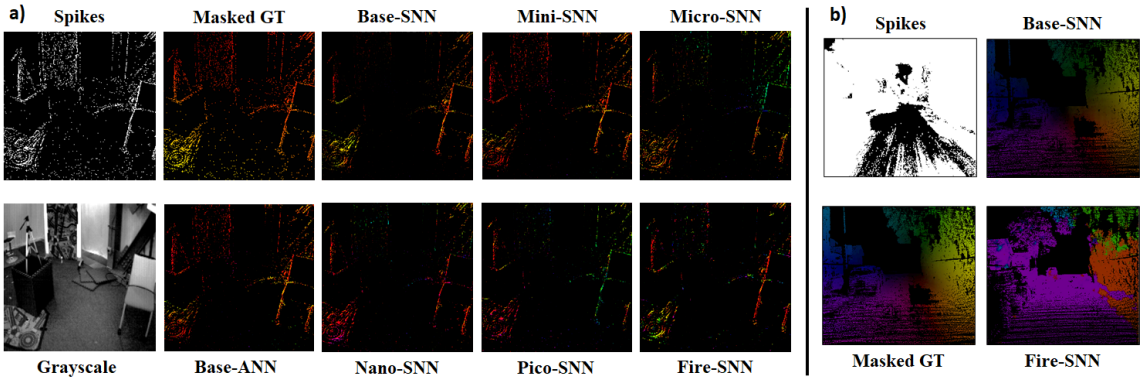

Fig. 4: a) Results on MVSEC using various SNN architectures. b) Results on DSEC-Flow Base-SNN and Fire-SNN.

*outdoor_day*1 and *indoor_flying*1,2,3 sequences for fair comparisons with past works [11], [12], [21], [26]. We also compare with other works [22], [27] that train on a separate high speed dataset (UZH-FPV Drone racing dataset [37]). However, these comparisons are not totally fair due to the drone racing dataset [37] having a much wider distribution of optical flow vectors than MVSEC leading to better learning. We train and evaluate on an event volume corresponding to a consecutive pair of grayscale images (dt=1).

During training, we perform random flips, rotations, and crop inputs to a 256×256 size. We use the Adam optimizer [38] and train for 100 epochs with a batch size of 8 and an initial learning rate of 10^{-4}. The learning rate is scaled by 0.7 every 10 epochs. The surrogate width γ for the inverse tangent in the LIF neuron is set to 10 and the smoothness loss weighting factor is also set to 10. The number of bins (B) was set to 10, leading to 5 timesteps (T) and approximately 2000 events per bin on average. These hyperparameters were chosen heuristically and based on past works.

2) DSEC-Flow: The DSEC-Flow dataset was released in conjunction with [15] adding to the original DSEC dataset [14]. It features VGA resolution event cameras and provides optical flow ground truths for 24 driving sequences for a total of 7800 training samples and 2100 test samples. However, the ground truths for the test sequences are not open-source, leaving us with only the training data. Thus, we create our own $80-20(\%)$ split for training and testing using the available 7800 samples corresponding to 18 sequences. We train for 50 epochs with random flips and a 288×384 crop. The learning rate is kept fixed at 10^{-4}.

Flow is evaluated using the average endpoint error (AEE) (also termed as endpoint error (EPE)) that measures the mean distance between the predicted and ground truth flow. This is done only for pixels containing events (m).

$$AEE = \frac{1}{m} \sum_{m} \left\| (u,v)_{\text{pred}} - (u,v)_{\text{gt}} \right\|_2 \tag{15}$$

B. Architectural Ablations

We analyze several smaller model sizes by reducing the number of base channels (b) in the original U-Net architecture. Namely the architectures are Base(64), Mini(32), Micro(16), Nano(8) and Pico(4), where the quantity in parentheses represents the base channels (b) for that model. All these architectures along with Fire-FlowNet-SNN are trained in an SSL fashion on the MVSEC dataset and in a supervised manner on the DSEC-Flow dataset. In addition, we also train their corresponding ANN models to have an iso-architecture comparison between SNN and ANN implementations. Note, that the SNN and ANN models are based on architectures explored in [12] and [11], respectively. For the ANN training, the event bins are passed as channels rather than as a sequence. Table. III shows the number of parameters in each model. Note that, even though Fire-FlowNet has the least number of parameters ($57K$), it incurs a hefty 3.7×10^9 operations due to no downsampling of activations.

C. Quantitative Results

The quantitative results are showcased in Table. I and Table. III and visualized in Fig. 4. From Table I we observe that Fusion-FlowNet outperforms all methods for nearly all sequences. This is mainly due to the usage of frame information in addition to events. The second best is [21], the unsupervised pipeline trained on MVSEC using the event-warping (E) loss. Our ANN models demonstrate respectable performance compared to prior works, and show a gradual increase in AEE with decreasing model size. The result of interest is that our fully-spiking models consistently outperform the corresponding ANN models (13% lower AEE on average) for all model sizes. In fact, our Base-SNN shows similar or better performance compared to the only other fully-spiking work (XLIF-EV-FlowNet [27]) even with the inferior intensity based (I) loss. Although, the corresponding FireNet model (XLIF-FireNet [27]) marginally outperforms our Fire-SNN model. This can be well attributed to the event-warping (E) loss and also the use of FireNet which contains explicit recurrence in the form of GRU units.

On DSEC-Flow, our work, although unable to outperform E-RAFT [15] which again uses explicit recurrence and iterative refinement, clearly outclasses EV-FlowNet [11] by $\sim 30\%$. E-RAFT is also evaluated on the original DSEC-Flow testset. Again, fully-spiking models show lower EPE $\sim 11\%$ (Base) and $\sim 17\%$ (Fire) compared to ANN models.

TABLE I: AEE results on MVSEC dataset. 1^{st} block: Mixed SOTA methods. 2^{nd} block: SSL methods using events and/or frames. 3^{rd} block: Our ANN models. 4^{th} block: Our fully-spiking SNNs. (E) Event-warping loss, (I) Photometric loss.

	Loss Type	outdoor_day1	indoor1	indoor2	indoor3	AEE Sum	Improvement (%)
EV-FlowNet [11]	I	0.49	1.03	1.72	1.53	4.77	-
ECN_{masked} [20]	E	**0.30**	-	-	-	-	-
Zhe et. al [21]	E	0.32	0.58	1.02	0.87	**2.79**	-
Back to Event $Basics_{Evf}$ [22]	E	0.92	0.79	1.40	1.18	4.29	-
Back to Event $Basics_{Fire}$ [22]	E	1.06	0.97	1.67	1.43	5.13	-
XLIF-EV-FlowNet [27]	E	0.45	0.73	1.45	1.17	3.8	-
XLIF-FireNet [27]	E	0.54	0.98	1.82	1.54	4.88	-
Spike-FlowNet [12]	I	0.49	0.84	1.28	1.11	3.72	-
Fusion-FlowNet [26]	I	0.59	**0.56**	**0.95**	**0.76**	2.86	-
Ours (Base-ANN)	I	0.48	0.84	1.59	1.36	4.27	-
Ours (Mini-ANN)	I	0.52	0.9	1.68	1.44	4.54	-
Ours (Micro-ANN)	I	0.57	0.95	1.74	1.48	4.74	-
Ours (Nano-ANN)	I	0.62	0.97	1.74	1.49	4.82	-
Ours (Pico-ANN)	I	0.65	1.04	1.78	1.53	5	-
Ours (Fire-ANN)	I	1.01	1.22	2.03	1.78	6.04	-
Ours (Base-SNN)	I	0.44	0.79	1.37	1.11	3.71	13.1
Ours (Mini-SNN)	I	0.46	0.83	1.4	1.17	3.86	14.9
Ours (Micro-SNN)	I	0.52	0.92	1.53	1.28	4.25	10.3
Ours (Nano-SNN)	I	0.52	0.93	1.54	1.31	4.3	10.7
Ours (Pico-SNN)	I	0.58	0.95	1.58	1.28	4.39	12.2
Ours (Fire-SNN)	I	0.72	1.08	1.8	1.47	5.07	**16.1**

TABLE II: Computational efficiency on MVSEC *(* #OPS for corresponding ANN)*

	#Parameters($\times 10^6$)	#OPS_{ANN}($\times 10^9$)	Avg. Spiking Activity(%)	#OPS_{SNN}($\times 10^9$)	$E_{Total}(mJ)$	Improvement ($\times$)
Ev-FlowNet [11]	13.04	5.34	-	-	24.54	1$\times$
Spike-FlowNet [12]	13.04	4.41	0.48	0.0158	20.29	1.21$\times$
$Fusion\text{-}FlowNet_{Late}$ [26]	7.55	2.85	0.147	0.0052	13.11	1.87$\times$
Ours (Base-SNN)	13.04	11.31*	45.83	25.92	23.3	1.05$\times$
Ours (Mini-SNN)	3.41	4.3*	45.15	9.71	8.75	2.8$\times$
Ours (Micro-SNN)	0.93	1.88*	61.9	5.81	5.25	4.66$\times$
Ours (Nano-SNN)	0.27	0.96*	55.55	2.67	2.4	10.2$\times$
Ours (Pico-SNN)	0.092	**0.57***	74.12	**2.11**	**1.9**	**12.9**$\times$
Ours (Fire-SNN)	**0.057**	3.7*	**28.57**	5.28	4.75	5.16$\times$

TABLE III: EPE results on DSEC-Flow dataset.

	EPE	1PE	2PE	3PE
EV-FlowNet [11]	2.32	55.4	29.8	18.6
E-RAFT [15]	**0.79**	**12.5**	**4.7**	**2.7**
Ours (Base-ANN)	1.82	22.5	9.8	5.1
Ours (Base-SNN)	1.62	19.2	8.4	4.5
Ours (Fire-ANN)	5.92	82.6	59.1	40.2
Ours (Fire-SNN)	4.88	66.8	37.5	24.9

D. Computational Efficiency

We validate the efficiency of our fully-spiking models in terms of the number of network parameters and inference energy. For estimating the inference energy, we utilize the estimation method used in [26], [39]. SNNs perform sparse ACcumulate (AC) operations due to their binary outputs, whereas ANNs perform dense Multiply-and-ACcumulate (MAC) operations. The energy required for MAC and AC operations are: E_{MAC}=4.6pJ and E_{AC}=0.9pJ, for a 32-bit floating-point computation in 45nm CMOS technology [40]. This makes the AC operation 5.1$\times$ more energy-efficient than MAC. The layer-wise synaptic operations in SNNs can be computed as the product of the #neurons (M_l), mean firing activity (F_l), #synaptic connections (C_l) and, #timesteps (T).

$$\#\text{OPS}_{\text{SNN}} = T\sum_l M_l C_l F_l,\ \#\text{OPS}_{\text{ANN}} = \sum_l M_l C_l \tag{16}$$

$$\text{E}_{\text{SNN}} = \#\text{OPS}_{\text{SNN}} \times \text{E}_{\text{AC}},\quad \text{E}_{\text{ANN}} = \#\text{OPS}_{\text{ANN}} \times \text{E}_{\text{MAC}} \tag{17}$$

Table II shows the computational efficiency of our proposed SNNs. Our SNN models show much higher spiking activity compared to hybrid SNN-ANNs [12], [26]. This is because the models learn to reduce the LIF thresholds and increase leak to overcome vanishing spikes, allowing for better performance while still being efficient. Our SNNs also incur much lower computational cost due to sparse event-driven and binary computations offering $1-2$ orders of magnitude improvement in compute energy. We observe that Nano-SNN offers an optimal tradeoff between performance (10.7% and 9.9% lower AEE) and efficiency (1.84$\times$ and 10.2$\times$ lower energy) than Nano-ANN and [11], respectively.

V. CONCLUSION

We propose a framework for training deep fully-spiking networks for optical flow estimation using adaptive neuronal dynamics. We show that our models capture timing information better making them suitable as low-latency and low-energy solutions for the resource-constrained edge while maintaining application accuracy. This work can be extended to other perception tasks, eventually fueling the controls and planning pipelines and enabling high-speed and collision-free robot navigation.

Acknowledgment: This work was supported in part by, Center for Brain-inspired Computing (C-BRIC), a DARPA sponsored JUMP center, Semiconductor Research Corporation (SRC), National Science Foundation, the DoD Vannevar Bush Fellowship, and IARPA MicroE4AI.

REFERENCES

[1] A. Borst, J. Haag, and D. F. Reiff, "Fly motion vision," *Annual Review of Neuroscience*, vol. 33, no. 1, pp. 49–70, 2010. PMID: 20225934.

[2] M. Srinivasan, S. Zhang, M. Lehrer, and T. Collett, "Honeybee navigation en route to the goal: visual flight control and odometry," *Journal of Experimental Biology*, vol. 199, no. 1, pp. 237–244, 1996.

[3] E. Baird, N. Boeddeker, M. R. Ibbotson, and M. V. Srinivasan, "A universal strategy for visually guided landing," *Proceedings of the National Academy of Sciences*, vol. 110, no. 46, pp. 18686–18691, 2013.

[4] J. R. Serres and F. Ruffier, "Optic flow-based collision-free strategies: From insects to robots," *Arthropod structure & development*, vol. 46, no. 5, pp. 703–717, 2017.

[5] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128× 128 120 db 15 μs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 566–576, Feb 2008.

[6] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240 × 180 130 db 3 μs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, pp. 2333–2341, Oct 2014.

[7] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C.-K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y.-H. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, pp. 82–99, January 2018.

[8] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, pp. 1537–1557, Oct 2015.

[9] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in neuroscience*, vol. 9, p. 437, 2015.

[10] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, *et al.*, "A low power, fully event-based gesture recognition system," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7243–7252, 2017.

[11] A. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "Ev-flownet: Self-supervised optical flow estimation for event-based cameras," in *Proceedings of Robotics: Science and Systems*, (Pittsburgh, Pennsylvania), June 2018.

[12] C. Lee, A. Kosta, A. Z. Zhu, K. Chaney, K. Daniilidis, and K. Roy, "Spike-flownet: Event-based optical flow estimation with energy-efficient hybrid neural networks," in *European Conference on Computer Vision*, pp. 366–382, Springer, 2020.

[13] A. Z. Zhu, D. Thakur, T. Özaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The multivehicle stereo event camera dataset: An event camera dataset for 3d perception," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2032–2039, 2018.

[14] M. Gehrig, W. Aarents, D. Gehrig, and D. Scaramuzza, "Dsec: A stereo event camera dataset for driving scenarios," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4947–4954, 2021.

[15] M. Gehrig, M. Millhäusler, D. Gehrig, and D. Scaramuzza, "E-raft: Dense optical flow from event cameras," in *2021 International Conference on 3D Vision (3DV)*, pp. 197–206, IEEE, 2021.

[16] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi, "Event-based visual flow," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, pp. 407–417, Feb 2014.

[17] T. Brosch, S. Tschechne, and H. Neumann, "On event-based optical flow detection," *Frontiers in neuroscience*, vol. 9, p. 137, 2015.

[18] G. Gallego, H. Rebecq, and D. Scaramuzza, "A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation," *CoRR*, vol. abs/1804.01306, 2018.

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[20] C. Ye, A. Mitrokhin, C. Fermüller, J. A. Yorke, and Y. Aloimonos, "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)*, pp. 5831–5838, 2020.

[21] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "Unsupervised event-based learning of optical flow, depth, and egomotion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 989–997, 2019.

[22] F. Paredes-Vallés and G. C. de Croon, "Back to event basics: Self-supervised learning of image reconstruction for event cameras via photometric constancy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3446–3455, 2021.

[23] T. Stoffregen, C. Scheerlinck, D. Scaramuzza, T. Drummond, N. Barnes, L. Kleeman, and R. Mahony, "Reducing the sim-to-real gap for event cameras," in *European Conference on Computer Vision*, pp. 534–549, Springer, 2020.

[24] H. Rebecq, D. Gehrig, and D. Scaramuzza, "Esim: an open event camera simulator," in *Conference on robot learning*, pp. 969–982, PMLR, 2018.

[25] F. Paredes-Vallés, K. Y. Scheper, and G. C. De Croon, "Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion perception," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 8, pp. 2051–2064, 2019.

[26] C. Lee, A. K. Kosta, and K. Roy, "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)*, pp. 6504–6510, IEEE, 2022.

[27] J. Hagenaars, F. Paredes-Vallés, and G. De Croon, "Self-supervised learning of event-based optical flow with spiking neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 7167–7179, 2021.

[28] N. Rathi and K. Roy, "Diet-snn: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[29] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2661–2671, 2021.

[30] B. Yin, F. Corradi, and S. M. Bohté, "Effective and efficient computation with multiple-timescale spiking recurrent neural networks," in *International Conference on Neuromorphic Systems 2020*, pp. 1–8, 2020.

[31] M. Mahowald, "The silicon retina," in *An Analog VLSI System for Stereoscopic Vision*, pp. 4–65, Springer, 1994.

[32] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu, "Spatial transformer networks," in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[33] L. F. Abbott, "Lapicque's introduction of the integrate-and-fire model neuron (1907)," *Brain research bulletin*, vol. 50, no. 5-6, pp. 303–304, 1999.

[34] J. Y. Jason, A. W. Harley, and K. G. Derpanis, "Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness," in *European Conference on Computer Vision*, pp. 3–10, Springer, 2016.

[35] D. Sun, S. Roth, and M. J. Black, "A quantitative analysis of current practices in optical flow estimation and the principles behind them," *Int. J. Comput. Vision*, vol. 106, pp. 115–137, Jan. 2014.

[36] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[37] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6713–6719, IEEE, 2019.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[39] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in neuroscience*, vol. 11, p. 682, 2017.

[40] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 10–14, IEEE, 2014.