2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-1714-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IROS51168.2021.9635975

Self-Supervised Optical Flow with Spiking Neural Networks and Event Based Cameras

Kenneth Chaney[1], Artemis Panagopoulou[1], Chankyu Lee[2], Kaushik Roy[2], and Kostas Daniilidis[1]

Abstract— **Optical flow can be leveraged in robotic systems for obstacle detection where low latency solutions are critical in highly dynamic settings. While event-based cameras have changed the dominant paradigm of sending by encoding stimuli into spike trails, offering low bandwidth and latency, events are still processed with traditional convolutional networks in GPUs defeating, thus, the promise of efficient low capacity low power processing that inspired the design of event sensors. In this work, we introduce a shallow spiking neural network for the computation of optical flow consisting of Leaky Integrate and Fire neurons.**

Optical flow is predicted as the synthesis of motion orientation selective channels. Learning is accomplished by Backpropagation Through Time. We present promising results on events recorded in real "in the wild" scenes that has the capability to use only a small fraction of the energy consumed in CNNs deployed on GPUs.

I. Introduction

Spiking Neural Networks (SNNs) and event based cameras (DVSs) have recently seen many advancements that have opened up the field of deep learning based computing models. In Zhu et al. [2], a dense input representation was introduced that maintained the temporal fidelity for a properly chosen set of events. This work allowed for accurate prediction of optical flow, depth and ego motion. SNNs have recently seen gradient approximation methods in multiple works (recently [3] and [4]). These surrogate gradient methods have enabled the training of models for classification tasks. The asynchronous nature of the sensor and computing paradigm enable SNNs and DVSs to work together to perform tasks that are similar to their biological counterparts.

Optical flow calculations are used in numerous downstream robotics tasks such as obstacle avoidance, UAV landing, UAV hovering, and odometry [5]. The importance of motion recognition is seen by the prevalence throughout the animal kingdom; Drosophila [6], Falcons [7], and humans [8]) all utilize this information for similar tasks which we task robots with.

In silicon, the DVS sensor [9] provides an approximation of retina cells, transmitting spike information only when the photoreceptor changes in intensity. When compared to standard cameras, which rely upon integrated frames, the DVS shows gains in power efficiency, bandwidth, latency, and dynamic range. The core drawback for most systems is the lack of decades of research in tasks such as feature tracking, optical flow, structure from motion, and object detection.

Recently, these downstream tasks have mainly been tackled on conventional CPU or GPU. Recent advances in leveraging deep learning to complete tasks such as optical flow [10], obstacle avoidance [11], [12], and image reconstruction [13], [14] perform at a high degree of accuracy. However, they fall behind in the power efficiency and asynchronicity of the DVS. Spiking Neural Networks (SNNs) have recently seen dedicated hardware emerge to address each of these needs. Most notably IBM TrueNorth [15], SpiNNaker [16], and Loihi [17] all provide a silicon based neuron simulation that can be configured to run functional subsets of a brain. These recent advances in silicon has allowed for small scale experiments such as SLAM [18] and error correction in path integration [19].

Supervised SNN training is difficult due to the lack of a gradient through the spike function itself. Surrogate gradient methods have allowed for approximations of the gradient through this function to be used to train networks. Methods such as SLAYER [3], require spikes to be present at every layer to construct a gradient that is non-zero. This is a requirement to start the learning process. The gradient approximation methods have been shown to work for classification tasks and rate encoding tasks, because solutions are able to be reached over time. Regression tasks such as optical flow from event based cameras have continuously changing outputs that do not allow for large amounts of time to elapse before resulting in an answer.

In this paper, we propose a novel optical flow pipeline that builds upon the recent advances in supervised SNN training as well as recent advances in methods training standard convolutional neural networks (CNN) for event based cameras. The technical contributions of this paper are as follows:

- Fully self-supervised convolutional SNN for optical flow from events.
- A novel vector output encoding scheme for spikes that minimizes the activity levels in the final layer.
- A novel architecture and pipeline that leverages multiple temporal delays, surrogate gradient methods, and rate-population encoding to perform dense optical flow estimation from events.
- Evaluation of low resolution and parameter count networks on MVSEC [1].

[1]University of Pennsylvania {chaneyk, artemisp, kostas} @seas.upenn.edu
[2]Purdue University {lee2216, kaushik} @purdue.edu

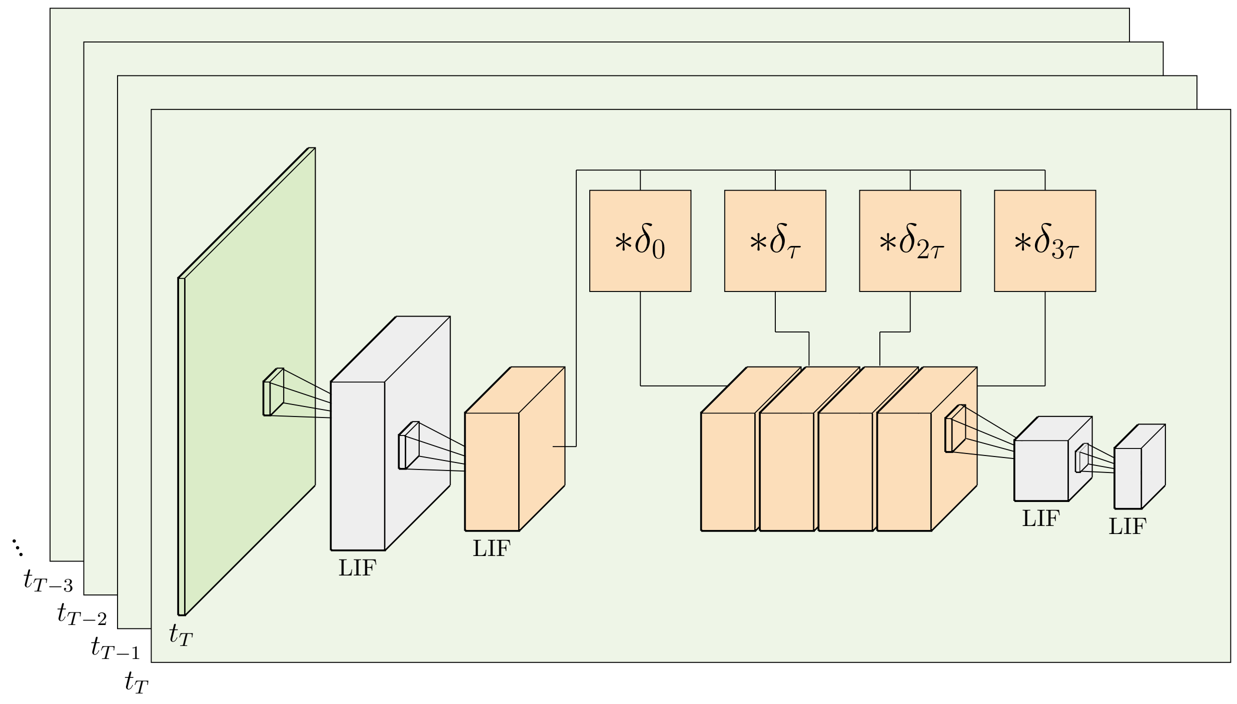

Fig. 1: Our network model consists of four populations of Leaky Integrate and Fire neurons that are connected by convolutional and synaptic delay connections. The synaptic delays allow for the network to observe what has happened in the past. These neuron populations contain state that evolves overtime as new inputs are fed into the network. The output populations are then decoded into optical flow, the decoding process can be seen in Figure 3.

II. Spiking Neural Networks

Their asynchronous nature and low power consumption make SNNs a natural match for sparse processing of temporal information. SNNs provide a frame-free alternative to ANNs (standard deep networks, either convolutional or fully connected) allowing dedicated hardware to exploit highly parallelizable and efficient temporal computations. The natural use of sparse temporal information allows for these networks to exploit the efficiencies internal to event based sensors such as DVS cameras [20], as they are not restricted by the temporal latency associated with frame-based methods [21].

SNNs encode temporal information for individual spikes, instead of solely keeping track of the firing rate of individual neurons; in this way SNNs can efficiently encode spatio-temporal information that would be lost through rate-encoding. In this section we discuss in more detail how SNNs are formalized with the aim to elucidate why they are a natural counterpart to event based sensors.

We employ a Leaky Integrate and Fire (LIF) model that provide basic computational functionality and modalities; tonic spiking, class 1 excitability, and integration [22]. These properties enable this neuron to respond proportionally to external stimulus as well as have memory of what recently occurred. The basic dynamics of a LIF neuron can be described as a simple resistor and capacitor in parallel [23]. The discrete time model used:

$$v[t] = v[t-1] - \frac{v[t-1] - v_{rest}}{\tau_m} + w_l * \delta_l * I_{input}[t] \quad (1)$$

The spike inputs from the previous layer can be delayed through δ_l and weighted through w_l. Output spikes are generated when $v[t] > \Theta$ which in turn resets $v[t]$. The similarities between SNNs and ANNs can be seen when the LIF model is abstracted and treated as any other non-linear layers common in deep learning (e.g., ReLU, sigmoid, Tanh). The advantage is that the temporal information is handled directly and can be manipulated to create more complex data paths through manipulation of δ_l. While there exist more complex and more complete models that estimate the functionality of physical neurons, in the context of deep learning the LIF model's simplicity enables the burden of complex behaviors to be spread out amongst whole populations.

III. Related Works

A. Event Based Cameras

Early DVS cameras consisted of low spatial resolution, 128×128 sensors [24], [20] that exemplified the ability of the temporal resolution to make up for the lack of spatial resolution. Recently the spatial resolution has been increasing up to the VGA [25] range.

Tracking egomotion of an event based camera can be done at low computational cost with a high effective frame

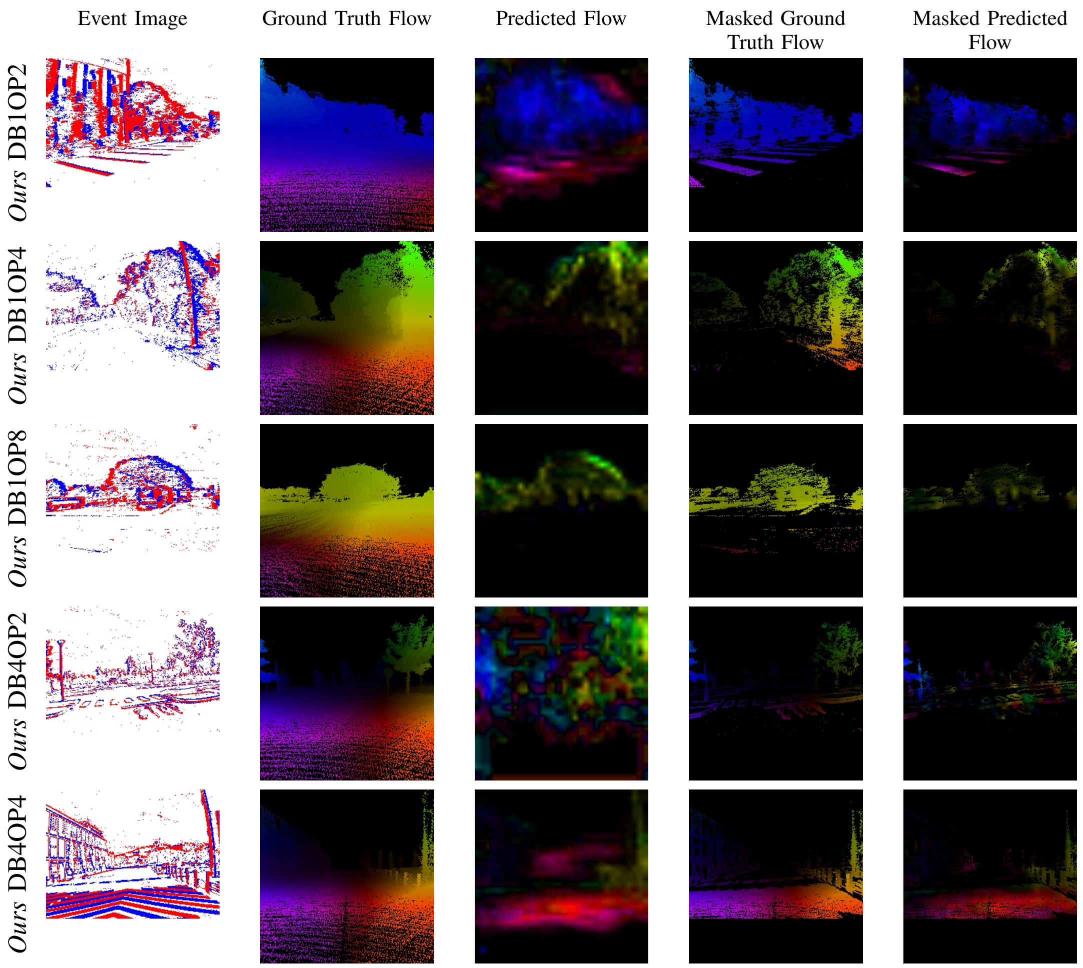

Fig. 2: Qualitative results are produced by running the networks over event streams from MVSEC [1]. The output of the spiking neural network is a stream of spikes over time. At every pixel, in x and y, we have a population of neurons that transmit updates through spikes. This information is decoded into optical flow by weighting each spike by the neuron's corresponding direction vector and averaging over the population and over time. The networks produce varying results in noisy regions, but once masked with the event image, this goes away.

rate. These works leverage depth maps [26], keyframes [27], and feature tracks [28]. While these works pull concepts from frame based cameras, they leverage the asynchronous features to solve issues of motion blur, allowing them to operate in high velocity scenarios.

Event based cameras naturally encode motion within a scene in the generated events. In a large number of cases, the events construct a local geometry that can be broken down into surfaces inside of the spatial temporal volume. This local geometry can be viewed as planes; local plane fitting [29] creates optical flow vectors that can track a wide variety of motion, but lacks in performance of corner cases (e.g., corners, circles). In contrast, leveraging optical flow methods from video processing for event based cameras has yielded results. Block matching [30] inspired by video compression methods becomes more efficient with meaningful binarized data. Combining an external high resolution sensor via a beam splitter allows for a best of both worlds approach to continuously estimate the optical flow gradient through time at a high spatial resolution [31].

Deep learning has been utilized to predict optical flow using synchronized image stream from the DAVIS sensors to construct the photometric loss with an event based input [10]. Zhu et al. [2] construct an event volume representation to maintain the temporal fidelity of the DVS sensor. The network is then trained purely on event data using a motion compensation loss proposed by Mitrokhin et al. [32].

Most relevant to this paper, Paredes et al. [33] proposed a method for unsupervised optical flow. While unsupervised, the network was tuned and trained for each scene shown. The learning implementented poses a challenge as it requires shared weights to be updated simultaneously. The problem arises because this weight-sharing is not natively supported on low power SNN processors such as the Loihi and instead would require a GPU to accompany a deployment.

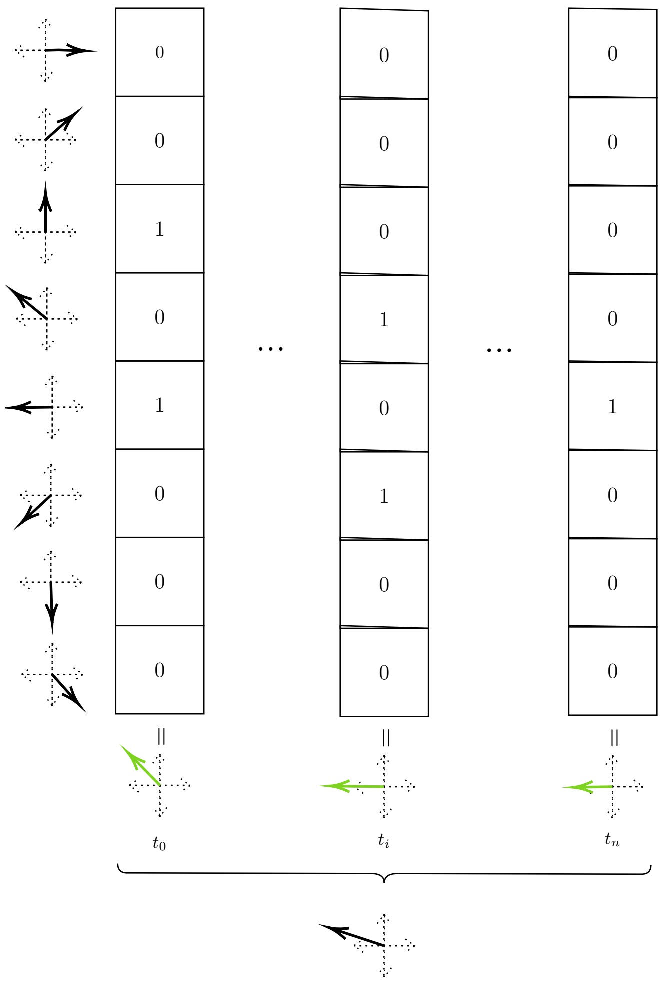

Fig. 3: The output of the spiking neural network is a stream of spikes over time. At every pixel, in x and y, we have a population of neurons that transmit updates through spikes. This information is decoded into optical flow by weighting each spike by the neuron's corresponding direction vector and averaging over the population and over time.

B. Spiking Neural Networks

The computational capacity of Spiking Neural Networks relies heavily upon the underlying computational model that describes the neurons. Izhikevich [22] discusses models ranging from Integrate and Fire, a simple model which is not capable of complex firing patterns, to Hodkin-Huxley, a complex biophysical model which exhibits most firing patterns observed in the brain.

Neftci et al. [4] and Shrestha et al. [3] examine approximating the gradient function of the spike operator to allow backpropagation through this non-differentiable function. Shrestha et al. focus on a function that assigns the errors appropriately through time, explicitly leveraging the temporal nature of SNNs. These methods have provided the basis for learning functions within the context of deep learning frameworks. Predicting angular velocity with an SNN [34] shows the a global regression problem. A hybrid SNN/CNN optical flow network [35] tackles the problem of splitting a network efficiently between two devices to product full resolution and high accuracy optical flow.

Hazan et al. [36] provide BindsNET, a general library for SNN simulation that leverages the PyTorch CPU and GPU computations. BindsNET only supports local learning rules (e.g., STDP [37]) and does not leverage the built in autograd internal to PyTorch.

Bekolay et al. [38] designed Nengo to leverage the Neural Engineering Framework, Semantic Pointer Architecture [39], and online local error rules to tackle dynamical and symbolic systems. SPAUN [40] was constructed through composing many subsystems that were designed to each perform specific tasks.

IV. Method

A. Layer communications

The forward pass, similar to recurrent neural networks (RNNs), requires inputs over time and receives outputs over time. In our case, both inputs and outputs are spikes which can be represented most compactly in the AER format on hardware (or software systems) that supports this format. However, PyTorch [41] only supports dense data representations for convolution operators, so a frame needs to be constructed for use during training. This can be done by placing the events at an instance in time, $\{e_i = (d_{0,i}, d_{1,i}, \dots, d_{n,i})\}_{i=0,1\dots M-1}$, into a frame:

$$F_t(d_0, d_1, \dots d_n) = \sum_{e_i \in (d_0, d_1, \dots d_n)} 1 \tag{2}$$

This frame is a quantized version of what is proposed by Zhu et al. [2]. However, instead of utilizing time information as channels, each time step receives the events in $[t_{i-1}, t_i)$. The primary methods for the network to retain prior information is to store in its neurons, through recurrent connections, or synaptic delays.

B. Backpropagation

The spike function that is central to SNNs, is non-differentiable at the point of interest. For backpropagation, a surrogate gradient allows for the errors to be correctly quantified in the forward pass and an approximate gradient used during the backward pass. In this work, the gradient of the fast sigmoid, with a scaling factor, is used as an approximation to the gradient of the heavy sided function, Neftci et al. [4]:

$$v[t-1] > \theta \tag{3}$$
$$v[t-1] - \theta > 0 \tag{4}$$
$$v_{-}[t-1] > 0 \tag{5}$$

After removing θ from immediate consideration, we can see that $v_{-}[t-1] > 0$ is the Heaviside function, $\mathbf{H}$, with respect to $v_{-}[t-1]$, thus the approximation of the derivative

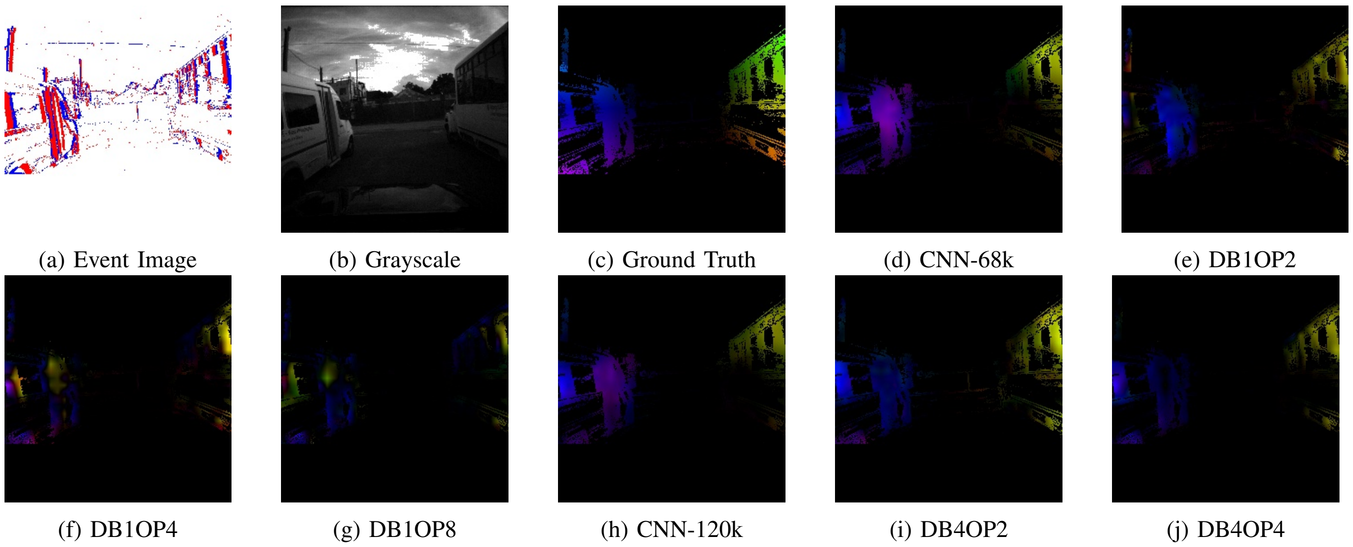
(a) Event Image (b) Grayscale (c) Ground Truth (d) CNN-68k (e) DB1OP2 (f) DB1OP4 (g) DB1OP8 (h) CNN-120k (i) DB4OP2 (j) DB4OP4

Fig. 4: Direct comparison of the separate networks in a challenging scene. The two CNNs tend to produce similar results and fail in similar ways. For the SNNs, the addition of the delay blocks tends to improve the worst case scenarios of the network. All images in this figure are masked with the event image to compare outputs where data is present.

is:

$$\frac{\partial L}{\partial v_{-}[t-1]} = \frac{\partial L}{\partial \mathbf{H}(v_{-}[t-1])}\frac{\partial \mathbf{H}(v_{-}[t-1])}{\partial v_{-}[t-1]} \tag{6}$$

$$\frac{\partial L}{\partial v_{-}[t-1]} \approx \frac{\partial L}{\partial \mathbf{H}(v_{-}[t-1])}\frac{1}{(|v_{-}[t-1]|+1)^2} \tag{7}$$

This approximation contains the same fundamental issues that the sigmoid contains as a non-linear operation in larger multi-layer networks, which is vanishing gradients. This is slightly mitigated through augmenting (7) with a scale factor hyper parameter, λ:

$$\frac{\partial L}{\partial v_{-}[t-1]} \approx \frac{\partial L}{\partial \mathbf{H}(v_{-}[t-1])}\frac{1}{(\lambda|v_{-}[t-1]|+1)^2} \tag{8}$$

Applying this approximation to a full simulation over many timesteps leads to this method being referred to as Backpropagation Through Time (BPTT). With each timestep being linked through the neuron's membrane potential, even if there are no explicit recurrent connections.

C. Delay blocks

SNNs are capable of manipulating input signals through not only the weight associated with each, but also delaying the signals in time. The temporal aspect of optical flow calculations are explored in energy based models [42]. In the context of network architectures, delay blocks are constructed to delay signals by a fixed amount of time. Combining multiple of these blocks enables the network to compare these distinct times explicitly. In addition, the delay blocks give the BPTT algorithm a shortcut to connect the spatial features through time in a more explicit context. These blocks are seen in Figure 1 as inline δ_τ blocks.

D. Output Representation

The coding of output spikes affect a network's ability to describe small or rapid changes in the input signal. The neural engineering framework (NEF) [43] provides mechanisms for constructing optimal population-temporal coding given an encoding into a set population size. The NEF inspires the output representation chosen, except instead of a given encoding, a desired decoding is given and the network learns an encoding that leverages this decoding. For an 2 dimensional $\{d_0 \in [-1,1], d_1 \in [-1,1]\}$ with a population of N neurons over time T:

$$\begin{aligned} d_0(x,y,t_0,t_T) &= \frac{1}{T}\sum_{t_0<t\le t_T}\sum_{n<N}\delta_{x,y,n}(t)\cos(2\pi\frac{n}{N}) \\ d_1(x,y,t_0,t_T) &= \frac{1}{T}\sum_{t_0<t\le t_T}\sum_{n<N}\delta_{x,y,n}(t)\sin(2\pi\frac{n}{N}) \end{aligned} \tag{9}$$

This can be visually seen in Figure 3, where each neuron is represented as a vector which gets summed up over the population and time.

For $N >= 4$, (9) provides a representation that allows for zero to be encoded with minimal activity. Alternatively utilizing just two neurons to represent this same signal, to cover the full range the output would have to be shifted:

$$\begin{aligned} d_0(x,y,t_0,t_1) &= \frac{1}{T}\sum_{t_0<t\le t_1}\sum_{n<N}\delta_{x,y,0}(t) - 0.5 \\ d_1(x,y,t_0,t_1) &= \frac{1}{T}\sum_{t_0<t\le t_1}\sum_{n<N}\delta_{x,y,1}(t) - 0.5 \end{aligned} \tag{10}$$

Representing $d_0 = 0$ or $d_1 = 0$ would then be represented as each neuron spiking half the time. While this isn't directly an issue, it does create readout issues on SNN hardware where it is more efficient to read out less information as the lack of spikes are implied and not directly read out.

E. Network Architecture

The balance between batch size and number of simulation timesteps drives the total network size that is trainable on a single GPU. This is a direct implication of BPTT as each timestep is linked and required for updates of all parameters. The network is trained using a 1ms temporal resolution with batches that span 100ms. A four layer, feed forward, convolutional network was chosen (as seen in Figure 1). The first and second layers construct spatial features. The third layer contains delay blocks which creates copies of the signals through time for the network to process; the convolution after these delay blocks enables spatial and temporal analysis directly embedded in the structure of the network. The final layer decodes the final feature into the chosen output representation. In our case, the optical flow at time t is a 2D vector that can be described as utilizing (9) or (10) taking t_{window} timesteps into account:

$$F(x,y,t) = \{d_0(x,y,t-t_{window}), d_1(x,y,t-t_{window})\} \tag{11}$$

Summation over time creates a delay in the result. Therefore, it is important to balance the amount of time to reach a result with a larger output population.

F. Losses

We use the photometric loss, $\mathcal{L}_{\text{photo}}$, on the images produced by MVSEC even though input to the network is events, as originally shown in [10]. For the smoothness loss, we use the robust Charbonnier loss [44] as in [2]. $N(\vec{x})$ is the 4-connected neighborhood around $\vec{x}$ and $\lambda_{\text{smoothness}}$ is a hyperparameter weight. These losses are shown in full below:

$$\mathcal{L}_{\text{photo}}(u,v;I_t,I_{t+1}) = \sum_{x,y} \rho\big(I_t(x,y) - I_{t+1}(x+u(x,y), y+v(x,y)\big) \tag{12}$$

$$\mathcal{L}_{\text{smoothness}} = \sum_{\vec{x}} \sum_{\vec{y}\in N(\vec{x})} \sqrt{(A(\vec{x}) - A(\vec{y}))^2 + \epsilon^2} \tag{13}$$

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{photo}} + \lambda_{\text{smoothness}}\mathcal{L}_{\text{smoothness}}$$

V. Experiments

Since these networks are trained through backpropagation, a large and reliable dataset is needed. EVFlowNet [10] successfully utilized MVSEC [1] to train a deep network to perform the task of optical flow. MVSEC comes with a per frame optical flow in addition to the event and images streams that are native to the DVS sensors. We train our networks on outdoor_day2; a longer sequence that has numerous independently moving objects. Our testing is done on outdoor_day1 which takes a unique path compared to the training sequence and has few moving objects which makes the ground truth flow more reliable for testing.

In all experimental results our network is labeled as *Ours*. The hyper parameters, output population and number of delay blocks, labeled with *OP#* and *DB#* respectively. The addition of the delay blocks changes the number of parameters drastically compared to those with fewer. For fair comparison two CNNs of similar size were trained, which was constructed with a 20 channel event volume as the input and a similar number of parameters. The network sizes can be seen in Table I; similar sized networks have been grouped together for easier comparison. CNN-68k and CNN-120k were constructed with 32 and 64 channels in intermediate layers respectively while all SNNs only project to 32 channels. The delay blocks account for the remainder of the network size for the larger SNNs. In all networks, the spatial resolution is reduced in the first three layers (through a stride of 2) and maintained in the 4th (through a stride of 1).

outdoor_day1	# Parameters	AEE (pixels)	Outliers (%)
CNN-68k	68,800	0.87	4.54
*Ours OP2DB*1	54,402	0.89	3.88
*Ours OP4DB*1	56,008	1.01	6.31
*Ours OP8DB*1	59,216	1.03	5.71
CNN-120k	120,000	0.89	4.11
*Ours OP2DB*4	117,604	**0.83**	**2.13**
*Ours OP4DB*4	119,208	0.89	3.58
Zhu et al. [2]	13,039,232	0.32	0.0

TABLE I: Quantitative analysis is done in a single shot manner to ensure that the SNN has no extra information available to it. Only the events within the time period in question are used. Networks are grouped by size with a reference CNN in each case. Zhu et al. [2] is the current state of the art model whose performance is given for comparison.

The comparison CNN is labeled as CNN, which was constructed with an event volume as the input representation and the same number of channels, from *Ours*, in each of the following layers for the closest comparison.

A. Average End-point Error

Evaluation of the accuracy is reported as Average End-point Error (AEE), the distance between the end points of the predicted and ground truth optical flow vectors:

$$AEE = \sum_{x,y} \left\| \begin{bmatrix} u(x,y)_{pred} \\ v(x,y)_{pred} \end{bmatrix} - \begin{bmatrix} u(x,y)_{gt} \\ v(x,y)_{gt} \end{bmatrix} \right\|_2 \tag{14}$$

In addition, outlier percentage, as laid out by Zhu et al. [10]. The networks trained, including the comparison CNN, produce optical flow at a substantially smaller resolution than those in which the input sensor or ground truth is available in. The decoded output from our network is $32 \times 32 \times 2$ and is upsampled to the original resolution, $256 \times 256 \times 2$, using bilinear interpolation. AEE is computed using the upsampled output to provide a per pixel estimate on the error. Table I contains the full listing of results for the CNN as well as the proposed SNN models. Figure 2 shows qualitative results from all networks in various times across the test sequence. Challenging scenes provide insight into the generalizability of the models; Figure 4 shows one such case.

outdoor_day1	Avg spike activity (%)	# Operations ($\times 10^8$)	Operation Type	Compute Energy Benefit ($\times$)	Avg output spike activity (%)	Avg. Bandwidth (MBps)
CNN-68k	-	3.94	MAC	1	-	0.8
*Ours OP2DB*1	**2.7**	**0.44**	AC	**45.7**	51.0	1.44
*Ours OP4DB*1	4.6	0.74	AC	27.5	0.6	**0.04**
*Ours OP8DB*1	**2.7**	0.45	AC	44.3	**0.3**	**0.04**
CNN-120k	-	5.25	MAC	1	-	0.8
*Ours OP2DB*4	6.7	1.16	AC	23.1	50.4	1.42
*Ours OP4DB*4	7.1	1.23	AC	21.9	1.2	0.07

TABLE II: The efficiency of the proposed network architecture shows large improvements in continuous operation costs. A CNN requires multiple discrete runs for every update, whereas a SNN provides sparse updates to each layer. These numbers are based off of providing 100Hz updates (10×1ms updates).

B. Continuous Execution

Continuous execution of optical flow networks is used in robotics contexts to better understand movement in the environment. An example setup would be to have an appropriate onboard accelerator that is designed for the type of network used (whether it is a CNN or SNN). Regardless, there are immutable components to this: first, the base computation of the accelerator must occur; second, communication of results back to the host system which is making the high level decisions. For this case, we analyze the networks producing 100Hz updates.

1) Computational Efficiency: SNNs benefit from both sparser and simpler calculations compared to ANNs. The number of synaptic operations is commonly used as a benchmark for the efficiency of neuromorphic hardware [45], [15], [46]. The number of operations for an ANN is straight forward to compute, $\sum_l M_l \times C_l$, where l is the layer index, M_l is the number of neurons at a given layer, and C_l is the number of connections at each layer. For SNNs, you must include the sparsity F_l at each layer and the number of time steps N in this calculation, giving you $N * \sum_l M_l \times C_l \times F_l$. To provide a 100Hz update, 10× 1ms time steps are needed.

Additionally, ANNs and SNNs have separate operators for the synaptic connections. For ANNs, multiply accumulate (MAC) computations are needed due to passing of real value numbers between the layers (a convolution multiplies floating point weights and values from previous layers). For SNNs, all that is needed is accumulate computations (AC) because the information that is passed between layers is binarized so the weights can be accumulated directly. AC computations are generally considered to be more efficient than MAC computations. In the case of 32-bit floating point numbers (45nm CMOS process), AC operations are $5.1\times$ more efficient than MAC operations [47].

Based on these two components, the efficiency of the proposed network is evaluated and reported in Table II.

2) Communications: Bandwidth in neural networks has become a bottleneck in large scale applications, during both training and inference. The current CNN implementations for optical flow require transfer of dense images back to the host for processing. Due to the sparse output representation, SNNs are able to transmit information with a low bandwidth and high temporal update rate. This maintains even when a real valued output is desired as an efficient output encoding scheme is able to compensate binary responses available at each time step.

In the tested CNN, the output frame results in a 32x32x2 image being transferred back to the host. To achieve an update rate of 100Hz, this would require a transfer of 8KB every 10 milliseconds (0.8MBps). In contrast, our SNN implementation constructs updates to the flow over time and only requires these updates to be reported. The standard AER packet size can be calculated through supplying an address in each dimension (x, y, channel), which results in $\sum_d \lceil log_2(shape[d]) \rceil$. Where d is an index into each dimension and $shape$ contains the shape of the output population. The sparsity, rate, and size of the update spikes are used to calculate the average bandwidth required by each method. Results of the bandwidth utilized by each model can be seen in Table II.

VI. Future Work

Moving forward, deployment on a dedicated SNN processor such as the Loihi will provide real world results for power consumption, scalability, and provide results given the other constraints that a physical implementation of a neuromorphic processor will have. This will also enable the integration with potential downstream neuromorphic algorithms such as SLAM [18].

VII. Acknowledgements

This work was supported in part by the Semiconductor Research Corporation (SRC) and DARPA, as well as the NSF-1703319 grant.

References

[1] 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.

[2] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, “Unsupervised event-based learning of optical flow, depth, and egomotion,” pp. 989–997, 2019.

[3] S. B. Shrestha and G. Orchard, “Slayer: Spike layer error reassignment in time,” in *Advances in Neural Information Processing Systems*, 2018, pp. 1412–1421.

[4] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks," *arXiv preprint arXiv:1901.09948*, 2019.

[5] H. Chao, Y. Gu, and M. Napolitano, "A survey of optical flow techniques for robotics navigation applications," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1, pp. 361–372, 2014.

[6] J. C. Tuthill, A. Nern, S. L. Holtz, G. M. Rubin, and M. B. Reiser, "Contributions of the 12 neuron classes in the fly lamina to motion vision," *Neuron*, vol. 79, no. 1, pp. 128–140, 2013.

[7] S. A. Kane and M. Zamani, "Falcons pursue prey using visual motion cues: new perspectives from animal-borne cameras," *Journal of Experimental Biology*, vol. 217, no. 2, pp. 225–234, 2014.

[8] F. P. Redlick, M. Jenkin, and L. R. Harris, "Humans can use optic flow to estimate distance of travel," *Vision research*, vol. 41, no. 2, pp. 213–219, 2001.

[9] L. Patrick, C. Posch, and T. Delbruck, "A 128x 128 120 db 15μ s latency asynchronous temporal contrast vision sensor," *IEEE journal of solid-state circuits*, vol. 43, pp. 566–576, 2008.

[10] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "Ev-flownet: Self-supervised optical flow estimation for event-based cameras," *arXiv preprint arXiv:1802.06898*, 2018.

[11] T. Stoffregen, G. Gallego, T. Drummond, L. Kleeman, and D. Scaramuzza, "Event-based motion segmentation by motion compensation," *arXiv preprint arXiv:1904.01293*, 2019.

[12] K. Chaney, A. Zihao Zhu, and K. Daniilidis, "Learning event-based height from plane and parallax," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[13] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza, "High speed and high dynamic range video with an event camera," *arXiv preprint arXiv:1906.07165*, 2019.

[14] ——, "Events-to-video: Bringing modern computer vision to event cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3857–3866.

[15] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[16] M. M. Khan, D. R. Lester, L. A. Plana, A. Rast, X. Jin, E. Painkras, and S. B. Furber, "Spinnaker: mapping neural networks onto a massively-parallel chip multiprocessor," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. Ieee, 2008, pp. 2849–2856.

[17] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[18] R. Kreiser, A. Renner, Y. Sandamirskaya, and P. Pienroj, "Pose estimation and map formation with spiking neural networks: towards neuromorphic slam," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2159–2166.

[19] R. Kreiser, Y. Sandamirskaya, *et al.*, "Error-driven learning for self-calibration in a neuromorphic path integration system," in *Robust Artificial Intelligence for Neurorobotics (RAI-NR Workshop 2019)*, 2019.

[20] T. Delbruck, "Frame-free dynamic digital vision," pp. 21–26, 2008.

[21] C. Farabet, R. Paz, J. Pérez-Carrasco, C. Zamarreño, A. Linares-Barranco, Y. LeCun, E. Culurciello, T. Serrano-Gotarredona, and B. Linares-Barranco, "Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel convnets for visual processing," *Frontiers in neuroscience*, vol. 6, p. 32, 2012.

[22] E. M. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE transactions on neural networks*, vol. 15, no. 5, pp. 1063–1070, 2004.

[23] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[24] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 x 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE journal of solid-state circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[25] M. Guo, J. Huang, and S. Chen, "Live demonstration: A 768$\times$ 640 pixels 200meps dynamic vision sensor," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–1.

[26] G. Gallego, J. E. Lund, E. Mueggler, H. Rebecq, T. Delbruck, and D. Scaramuzza, "Event-based, 6-dof camera tracking from photometric depth maps," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 10, pp. 2402–2412, 2017.

[27] H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization." in *BMVC*, 2017.

[28] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based visual inertial odometry," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 5816–5824.

[29] 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, no. 2, pp. 407–417, 2013.

[30] M. Liu and T. Delbruck, "Block-matching optical flow for dynamic vision sensors: Algorithm and fpga implementation," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4.

[31] M. M. Almatrafi and K. Hirakawa, "Davis camera optical flow," *IEEE Transactions on Computational Imaging*, pp. 1–1, 2019.

[32] A. Mitrokhin, C. Fermüller, C. Parameshwara, and Y. Aloimonos, "Event-based moving object detection and tracking," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[33] F. Paredes-Vallés, K. Y. W. Scheper, and G. C. H. E. 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*, 2019.

[34] M. Gehrig, S. B. Shrestha, D. Mouritzen, and D. Scaramuzza, "Event-based angular velocity regression with spiking networks," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4195–4202.

[35] C. Lee, A. K. 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*. Springer, 2020, pp. 366–382.

[36] H. Hazan, D. J. Saunders, H. Khan, D. T. Sanghavi, H. T. Siegelmann, and R. Kozma, "Bindsnet: A machine learning-oriented spiking neural networks library in python," *Frontiers in neuroinformatics*, vol. 12, p. 89, 2018.

[37] H. Markram and B. Sakmann, "Action potentials propagating back into dendrites trigger changes in efficacy of single-axon synapses between layer v pyramidal neurons," in *Soc. Neurosci. Abstr*, vol. 21, no. 3, 1995, p. 2007.

[38] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. C. Stewart, D. Rasmussen, X. Choo, A. Voelker, and C. Eliasmith, "Nengo: a python tool for building large-scale functional brain models," *Frontiers in neuroinformatics*, vol. 7, p. 48, 2014.

[39] C. Eliasmith, *How to build a brain: A neural architecture for biological cognition*. Oxford University Press, 2013.

[40] T. Stewart, F.-X. Choo, and C. Eliasmith, "Spaun: A perception-cognition-action model using spiking neurons," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 34, no. 34, 2012.

[41] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[42] T. Brosch, S. Tschechne, and H. Neumann, "On event-based optical flow detection," *Frontiers in Neuroscience*, vol. 9, p. 137, 2015. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2015.00137

[43] C. Eliasmith and C. H. Anderson, *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press, 2004.

[44] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud, "Two deterministic half-quadratic regularization algorithms for computed imaging," in *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, vol. 2. IEEE, 1994, pp. 168–172.

[45] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling spike-based backpropagation for training deep neural network architectures," *Frontiers in neuroscience*, vol. 14, 2020.

[46] 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.

[47] 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)*. IEEE, 2014, pp. 10–14.