Image Reconstruction from Neuromorphic Event Cameras using Laplacian-Prediction and Poisson Integration with Spiking and Artificial Neural Networks

Hadar Cohen Duwek Albert Shalumov Elishai Ezra Tsur*

Neuro-Biomorphic Engineering Lab (NBEL)
Department of Mathematics and Computer Science, The Open University of Israel

*elishai@nbel-lab.com

Abstract

Event cameras are robust neuromorphic visual sensors, which communicate transients in luminance as events. *Current paradigm for image reconstruction from event data relies on direct optimization of artificial Convolutional Neural Networks (CNNs). Here we proposed a two-phase neural network, which comprises a CNN, optimized for Laplacian prediction followed by a Spiking Neural Network (SNN) optimized for Poisson integration. By introducing Laplacian prediction into the pipeline, we provide image reconstruction with a network comprising only 200 parameters. We converted the CNN to SNN, providing a full neuromorphic implementation. We further optimized the network with Mish activation and a novel convoluted CNN design, proposing a hybrid of spiking and artificial neural network with < 100 parameters. Models were evaluated on both N-MNIST and N-Caltech101 datasets.*

1. Introduction

Some of the first and greatest successes in neuromorphic computing architectures have been in vision and sound processing [1]. Most neuromorphic vision sensors communicate transients in luminance via the Address Event Representation (AER) protocol. They are comprised of an array of silicon neurons; each generates spikes in response to a change in luminance in one particular location (or pixels). Spikes are time-multiplexed over an asynchronous data bus via an address encoder, which designates each spike with a corresponding address (usually, the neuron's x-y pixel coordinate). These frame-less and event-driven neuromorphic Dynamic Vision Sensors (DVSs) can resolve thousands of frames per second, have a fine temporal resolution, high dynamic range, no motion blur, and high signal-to-noise ratio. Moreover, since DVSs perform sensor-level data compression, they optimize data transfer, storage, and processing [2].

Recent works have narrowed the gap between conventional frame-based computer vision and that of an event-driven camera by using Convolutional Neural Networks (CNNs), which were optimized to reconstruct natural videos from events. For example, by utilizing a 10M parameters CNN termed U-net [3], Scaramuzza and colleagues reconstructed a video from its events, achieving state-of-the-art results [4] [5]. Recently, combined with recurrent connections and residual blocks, a CNN was used to reconstruct images with a smaller number of parameters, demonstrating a fast, lightweight network with only a minor drop in performance [6].

Here we proposed a Neural Engineering Framework (NEF)-based Spiking Neural Network (SNN) [7] for image reconstruction from event cameras, demonstrating a complete neuromorphic (brain-inspired) process. The Neural Engineering Framework (NEF) is one of the most utilized theoretical frameworks in neuromorphic computing. It brings forth a theoretical framework for a neuromorphic encoding, decoding, and transformation of mathematical constructs with spiking neurons, allowing the implementation of functional large-scale neural networks [8]. NEF was used to design a broad spectrum of neuromorphic frameworks ranging from robotic control [9] and visual processing [10] to perception [11]. It serves as the foundation for Nengo, a Python-based "neural compiler," which translates high-level descriptions to low-level neural models [12]. A version of NEF was compiled to work on both analog and digital neuromorphic circuitry [13], including the TrueNorth [14], the Loihi [15], the NeuroGrid [16], and the SpiNNaker [17].

In our proposed framework, events are driven into a convolutional SNN for processing, enabling process execution by neuromorphic hardware, considered energy-efficient [18]. Using Poisson integration [19] to reconstruct the image's intensity from its Laplacian, we demonstrate a reduced number of trainable parameters. Furthermore, we propose an even more compact non-spiking CNN, with Mish activation [20], achieving adequate image reconstruction with less than 100 parameters. We demonstrated our approach using the N-MNIST and N-Caltech101 datasets [21].

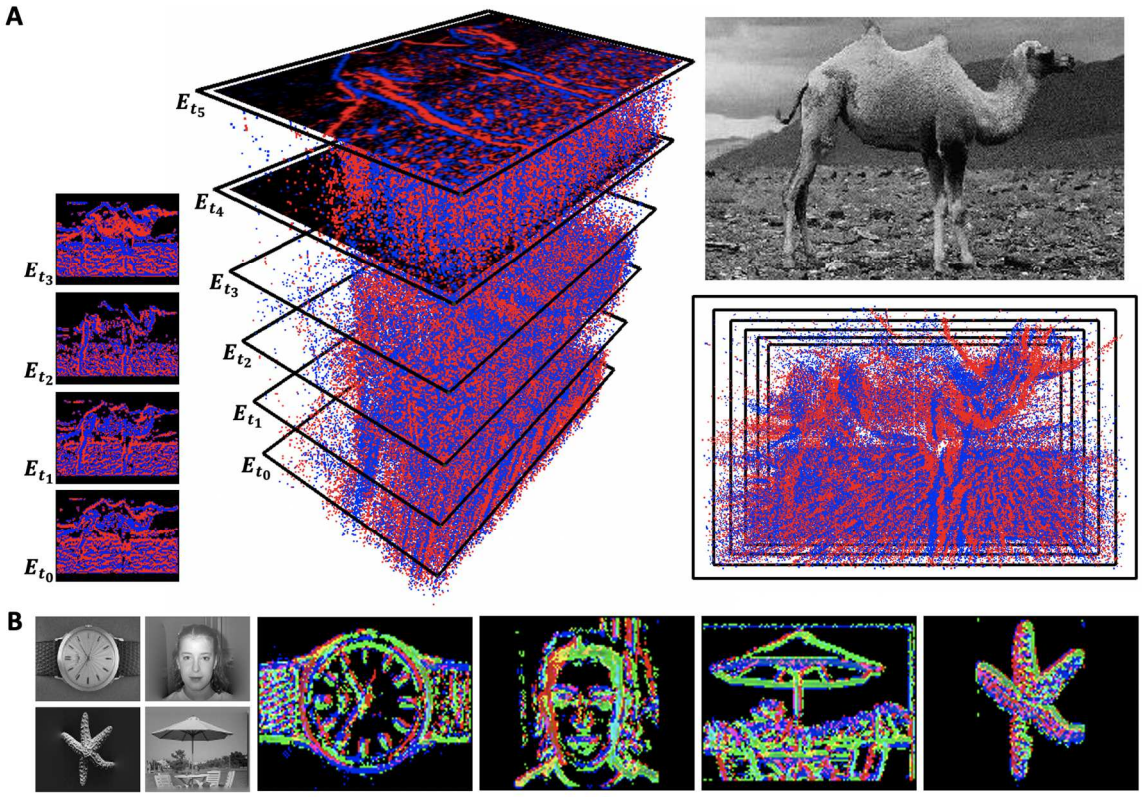

Figure 1. Visualization of the model's inputs. (A) An example image from the N-Caltech dataset and its corresponding events and event-frame tensors E; (B). Event frame tensors stacked and imaged by averaging and coloring each two consecutive frames (1st and 2nd frames averaged and visualized as red, the 3rd and 4th as green, and the 5th and 6th as blue).

2. Related works

Poisson image reconstructions are commonly used to reconstruct an image from its gradients [22]. Previous works solved the events-to-intensity reconstruction problem using an estimated gradient map of the image and integrated the gradients via Poisson integration. For example, Kim and colleagues used an extended Kalman filter to estimate a 2D gradient image from a rotating event camera. They used the gradient image to reconstruct the image by Poisson integration [23] [24]. Rebecq and colleagues extended the algorithm to cope with 3D scenes and 6-DoF motion [25]. Barua and colleagues also proposed reconstructing images from events by optimizing a sparse patch-based dictionary to match event patches with gradient patches using a simulator [26]. Their approach also utilized Poisson integration for image reconstruction.

Previous studies applied adversarial learning, through the use of Generative Adversarial Networks (GANs), to reconstruct High Dynamic Range (HDR) imagery from events [27] [28] [29]. GANs comprise a generator and a discriminator. The generator (usually composed of a U-Net architecture) aims to reconstruct an intensity image from events, which the discriminator cannot distinguish from the Ground Truth (GT) intensity image. Although GANs are unstable and hard to train [30], they have provided state-of-the-art performance on real-world event camera-generated datasets. Recently, Mohammad and colleagues proposed an intricate architecture, which involves optical flow estimation, feature enhancement, and super-resolution networks for image reconstruction in super-resolution and HDR [31]. These networks, however, are not based on SNNs and entail high parameter space.

Several works used events from event-based vision sensors as inputs to neuromorphic hardware-implemented SNN. For example, Riccardo and colleagues used Loihi-implemented SNN for event-driven gesture recognition [32]; Osswald and colleagues used SNNs to neuromorphically solve the stereo correspondence [33]; Jiand and colleagues used event-driven SNNs for object tracking [34], Seifozzakerini and colleagues used them for line detection [35], and Ezra Tsur and colleagues used evets to elucidate optical flow [10]. They were not used, however, for image reconstruction.

Here, we aim to solve the event-to-intensity reconstruction task on both N-MNIST and N-caltech101 event-based datasets with an SNN. Algorithm was based on a CNN optimized for Laplacian Prediction, which was

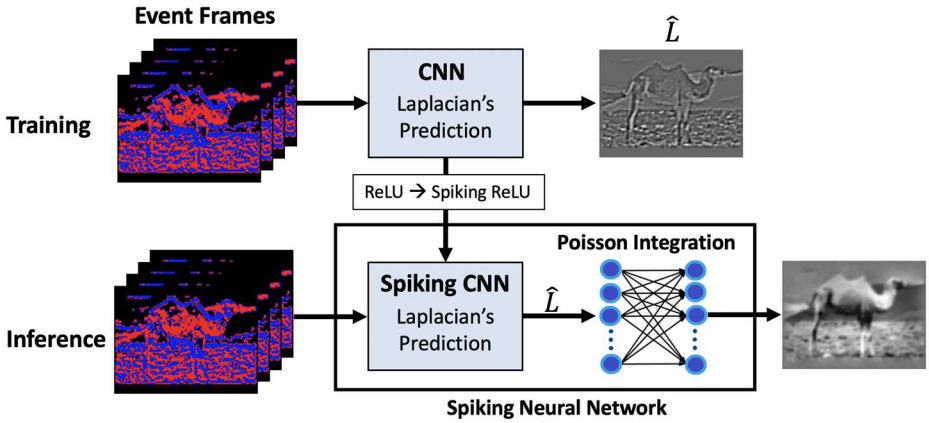

Figure 2. SNN architecture. At training time, CNNs are train to predict the Image's Laplacian. The trained CNNs are converted to SNNs for inference. At inference time, the predicted Laplacian is driven to a SNN, which performs Poisson integration, providing a fully spiking implementation of image reconstruction from event camera.

transferred to an SNN for inference. Inferred image Laplacian was introduced to a second SNN, optimized to solve the Poisson equation image reconstruction (filling-in SNN). We demonstrate that:

- CNN, optimized for Laplacian prediction, following the Poisson integration neural network, enable image reconstruction with compact parameter space.
- Image reconstruction can be entirely realized using a spiking CNN and a feedforward SNN, which can be implemented on neuromorphic hardware.
- Image reconstruction can be achieved with a neural network with shared-event filters CNN and Mish activation with less than 100 parameters.

3. Methods

3.1. Input representation and pre-processing

N-MNIST and N-Caltech101 are preprocessed as follows: each event-camera file was converted to an event-frame tensor E, which has the dimensions $H \times W \times T$, where H and W were shaped to 34 and T, the number of event-frames constituting each image in the dataset, is 90 and 120 for the N-MNIST and N-Caltach101 datasets, respectively. The N-Caltach101 was down-sampled from 180×240 pixels. E is defined using:

$$E(x, y, i) = \sum_{t=i\cdot\Delta t}^{(i+1)\cdot\Delta t} p_t, \tag{1}$$

where $\Delta t = 50$ mSec, $p_t \in (-1,1)$ is the polarity (increasing or decreasing luminance change event) of the event in time step t, $p \in (-1,1)$. Furthermore, each event frame is pre-processed by applying a spatial median filter, reducing noise with a 3x3 kernel. Input representation and pre-processing are described in **Figure 1.**

3.2. CNN Laplacian Prediction

Our approach initiates with a CNN, which predicts the Image's Laplacian given frame tensors, followed by a SNN optimized for Poisson Integration (**Figure 2**). We propose a five-layer CNN, where the first two comprise single-strided 3x3 convolutions (without down-sampling), and the last three layers comprise a single-strided 1x1 convolution kernel.

We used the Mean Absolute Error (MAE) loss in conjunction with the Structural Similarity Index Measure (SSIM) loss [36] and an Edge loss [37]:

$$\begin{aligned} Loss = {} & \lambda_1 \cdot MAE(L, \hat{L}) \\ & + \lambda_2 \cdot \left(1 - SSIM\left(PI(L), PI(\hat{L})\right)\right) \\ & + \lambda_3 \cdot Edge\_Loss\left(PI(L), PI(\hat{L})\right), \end{aligned} \tag{2}$$

where $\lambda_1 = 1$, $\lambda_2 = 0.25$, $\lambda_3 = 0.25$, L and $\hat{L}$ are the actual image and predicted Laplacians, respectively, and PI is the Poisson integration algorithm [22]. Note that while MAE loss was calculated directly from actual and predicted Laplacians, SSIM and edge loss were calculated from the reconstructed and predicted images. Edge loss allows the preservation of edges in the reconstructed image by considering edge similarity between the ground-truth intensity image and the reconstruction. We derive edges by calculating the squared sum of the image's derivatives in both vertical and horizontal directions. Derivative maps were thresholded to create binary maps. Finally, we use mean binary cross-entropy between the two maps to calculate the edge loss. Neurons' activations were defined as Rectified Linear Units (ReLU) [38]. The CNN was implemented using Keras [39].

We divided the N-caltech101 dataset sequences into 6097 training event sequences for network training, 1306 validation event sequences, and 1306 testing event

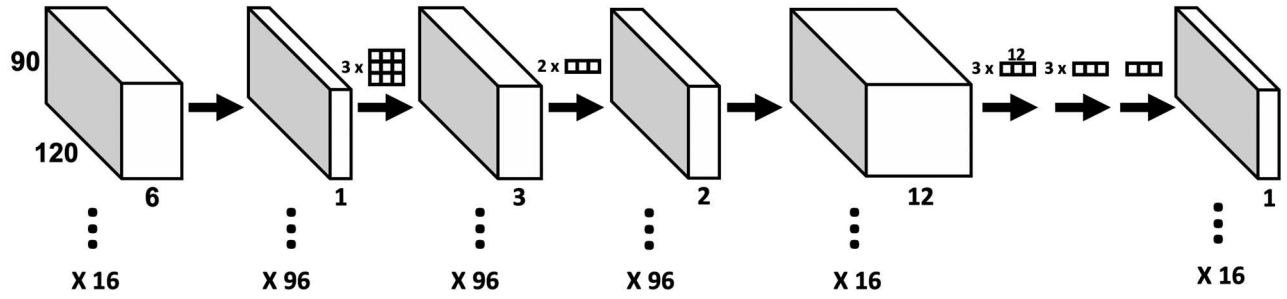

Figure 3. Shared-event filters CNN architecture. A < 100 parameters CNN for image reconstruction from event data.

sequences. We use Adam optimizer [40], with a batch size of 16 and an initial learning rate of 0.005. We set a 20% learning rate scheduling when reaching a plateau, with a minimum value of $2 \cdot 10^{-6}$. We consider training plateau, after six epochs, with non-improving validation loss, with:

$$Best_{new} < Best_{old}(1 - \alpha), \tag{3}$$

where α is the minimal required relative improvement (we use α=0.005). We stop training when mean validation SSIM has not improved over 20 epochs, with:

$$Best_{new} < Best_{old} - \Delta, \tag{4}$$

where Δ is the minimal required absolute improvement (we use Δ=0.005). We set the earliest stop epoch to 150.

3.3. CNN to SNN conversion

The trained CNN was converted to SNN using the Neural Engineering Framework (NEF)-based NengoDL library [41]. Here, ReLU activations were converted to Spiking rectified linear activation in the SNN, with which neurons' firing rate is proportional to positive input. Neural activity is rectified at zero. This spiking activation scheme is defined using a synaptic time constant (specifying a low-pass filter) and a maximal firing rate. Here we used a synaptic time constant of 10 mSec and a maximal firing rate of 100 and 5,000 Hz. Due to the temporal nature of SNN, each input image was presented to the model for 100 mSec. Spiking CNNs are used as differentiable approximations of their non-spiking versions. Network architecture is visualized in **Figure 3**.

3.4. Filling-in SNN

To neuromorphically implement Poisson integration, we designed a feedforward SNN. By using a finite difference numerical method, the Poisson equation can be rewritten as a linear system [42]:

$$A\vec{u} = \vec{b}, \tag{5}$$

where U is the image to be reconstructed, $\vec{u}$ is a column vector representing the pixels of the image U arrange in a natural ordering and A is the Laplace matrix defined as:

$$\mathrm{A} = \begin{bmatrix} D & -I & 0 & & 0 & 0 & 0 \\ -I & D & -I & \cdots & 0 & 0 & 0 \\ 0 & -I & D & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & D & -I & 0 \\ 0 & 0 & 0 & \cdots & -I & D & -I \\ 0 & 0 & 0 & & 0 & -I & D \end{bmatrix},$$

where I is the identity matrix, and D is defined as:

$$\mathrm{D} = \begin{bmatrix} 4 & -1 & 0 & & 0 & 0 & 0 \\ -1 & 4 & -1 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 4 & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & 4 & -1 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 4 & -1 \\ 0 & 0 & 0 & & 0 & -1 & 4 \end{bmatrix}.$$

The size of A depends on the image dimension. Here, we assume a fixed W matrix, where $W = A^{-1}$. The reconstructed image vector $\vec{u}$ can be calculated using:

$$\vec{u} = W\vec{b} \tag{6}$$

This solution was implemented as an SNN by connecting two neuron ensemble layers with a weight matrix W. Notably, W is calculated once for all the inputs, and it is not trainable, thus dramatically reducing the number of the network's parameters. The dimensions of W for input with spatial dimension $n \times m$ are $nm \times nm$.

3.5. Direct reconstruction

As our network design comprises two parts for image reconstruction: one for Laplacian prediction and the other for filling-in, we further compared it to a similar-size CNN, optimized for direct reconstruction (event-data to a reconstructed image). To train these networks, we modified the loss function, specified in **Equation** 2 as follows:

$$\begin{aligned} Loss = {} & \lambda_1 \cdot MAE(I, \hat{I}) \\ & + \lambda_2 \cdot (1 - SSIM(I, \hat{I})) \\ & + \lambda_3 \cdot Edge\_Loss(I, \hat{I}), \end{aligned} \tag{7}$$

where $\lambda_1 = 1, \lambda_2 = 0.25$ and $\lambda_3 = 0.25$, I is the actual image, and $\hat{I}$ is the reconstructed image.

3.6. Shared-event filters CNN

We utilized a novel CNN design with which the number of trainable parameters is further diminished to < 100 by curing the input as a video-like signal. The input batch is of dimension $B \times H \times W \times T$, accounting for batch size, height, width, and the number of frames, respectively. Inputs are reshaped and transposed to $(B * T) \times H \times W$ dimension and processed with a set of C convolutional filters. Following this phase, the $(B * T) \times H \times W \times C$ data is reshaped and transposed again, resolving with a $B \times H \times W \times (T * C)$ tensor. A second set of convolutions is applied, with a final single convolutional layer, resulting in a $B \times H \times W$ tensor. While the early layers can be thought of as performing preprocessing at the frame level, the second stage combines these preprocessed frames to compute the Laplacian. See Discussion for further details.

4. Results

To evaluate our proposed framework, we trained six five-layers Laplacian predicated CNNs with varying width constituting a different number of parameters (see **Table 1** for further details). Model #5, which comprises only 277 trainable parameters, was converted to SNN to evaluate a full neuromorphic pipeline. Predicted Laplacians were driven to the Filling-in SNN, which was described above.

4.1. Reconstruction via Laplacian predication

Table 1 summarizes CNN architectures and their corresponding performance measured on the test set for each network. Notably, the network's number of parameters is not directly correlated to performance as a smaller network (model #4) outperformed more extensive networks. We measured Peak signal-to-noise ratio (PSNR), SSIM, and Mean Square Error (MSE).

Predicted intensity images for the N-MNIST dataset from CNN model #6 are shown in **Figure 4.** Examples of predicted intensity images for the N-Caltech101 images from three selected CNN models (#1, #4, #5) are shown in

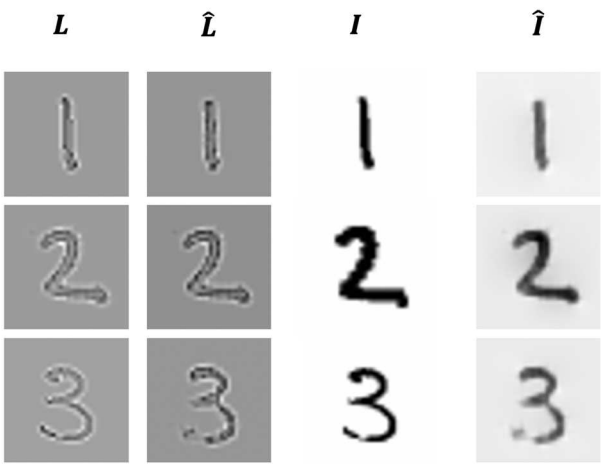

Figure 4. N-MNIST results. L , $\hat{L}$, I , $\hat{I}$ are the Ground Truth (GT) Laplacian, the predicted Laplacian, The GT image, and the reconstructed image, respectively.

Figure 5. Our results show that although having only 1,691 parameters, model #4 has comparable performance to model #1, which features 53,941 parameters. We demonstrate reasonable results with networks comprising only ≈ 200 parameters (models #5 and #6).

To demonstrate a fully spiking architecture, we converted model #5 (due to its relatively small number of trained parameters and high performance (**Table 1**)) to an SNN as was described earlier. Selected results are presented in **Figure 5** (model 5 SNN). As expected, while providing adequate results with a maximal firing rate of 100 Hz, the spiking version of model 5 is noisier than its non-spiking version. These results are dramatically improved when the maximal firing rate is set higher. As shown in **Figure 5** and **Table 1**, a 5,000 Hz SNN is comparable to its non-spiking version. We note, however, that a high spiking rate takes a toll on the system's energy efficiency.

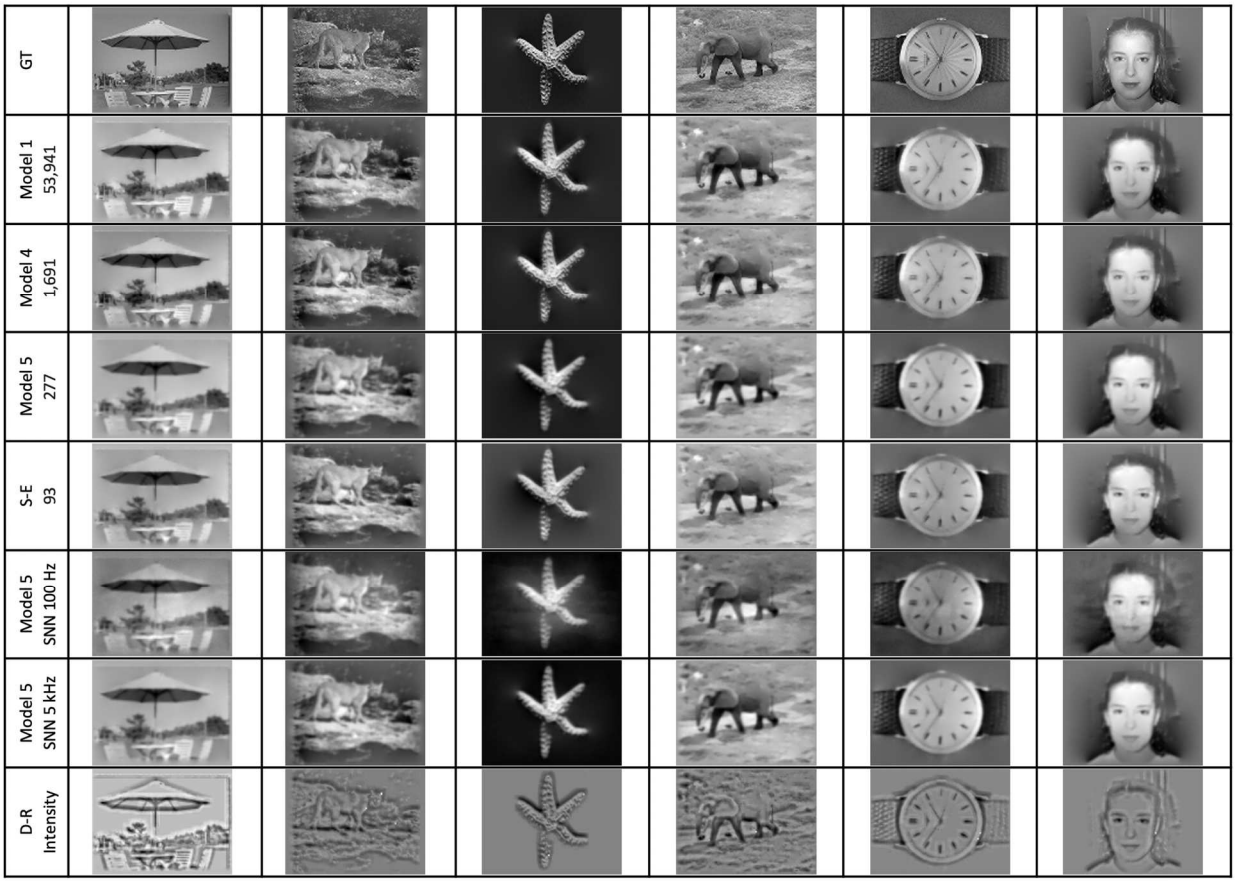

Figure 5. Selected image reconstructions. GT stands for ground truth (for reference); models #1, #4 and #5 stands for the neural networks, which feature Laplacian prediction followed by Poisson integration (reconstruction via Laplacian prediction); S-E stands for the Shared-events filters CNN; and D-R stands for the direct reconstruction CNN.

4.2. Direct reconstruction

We further compared our reconstruction via Laplacian prediction neural network to a similar-size CNN, optimized for direct. With direct reconstruction, there is no need for a Poisson integration process. We used CNNs models #1, #4, and #5 to directly reconstruct the image from the event tensors (see Methods).

Results are summarized in **Table 1**. Model #1 (the largest) achieved the best performance among the direct prediction models. Results show that with the specified loss function (**Equation 7**) and a 5-layers CNN, direct reconstruction only predicts the image's edges. Selected reconstructed images are shown in **Figure 4**. Notably, even though model #1 comprises > 50,000 parameters, a 205 parameters model (#6) when Laplacian prediction is involved outperforms it.

4.3. Shared-events filters CNN

To further optimize the number network's parameters, we designed a smaller, more intricate architecture, achieving adequate results with < 100 parameters (see Methods for further details). We tested this architecture with both ReLU and Mish activations. Mish is a smooth nonmonotonic function defined using $f(x) = x \cdot \tanh(\ln(1 + e^x))$. Mish was recently shown to outperform traditional activations in various cases, probably due to its positively unbounded, negatively bounded, smooth, and nonmonotonic characteristics [20].

Selected reconstructed images are shown in **Figure 4**. Even though this network is small, its performance is comparable to other, much larger networks (**Table 1**).

5. Discussion

We introduced CNNs for image reconstruction from event cameras via the prediction of the image's Laplacian and Poisson integration solved with an SNN. As Poisson integration is solved with an SNN, which does not require any learning procedure, the number of trainable parameters is dramatically improved. We further converted the CNNs to SNNs for a complete neuromorphic design. Our results demonstrate that conventional CNNs, with a low number of parameters, without a U-net, not auto-encoders, successfully reconstructs images from the N-Caltech101 and N-MNIST dataset. We show that while conventional simple-layered CNN, optimized for direct image reconstruction from events, can only reconstruct the image's edges, it can be used to elucidate the image Laplacian reasonably accurately. Thus, when used in conjunction with a filling-in neural network, it can be used to create an efficient neural network from image reconstruction.

While the Poisson integration SNN does not require training, it entails resolving a large matrix with nine constant parameters. This SNN can be further improved by dividing the event frames into patches, independently predicting and reconstructing them, finally, stitching them together to produce a full reconstructed image.

We further demonstrate our Laplacian prediction- and Poisson integration-based reconstruction with a dramatically reduced number of parameters by utilizing a novel CNN design and Mish activation. We demonstrate adequate image reconstruction with < 100 parameters CNN. This network reduces the number of parameters by curing the input as a video-like signal where each non-spatial dimension corresponds to a time slice. Most models use 2D convolutions as the first processing stage. 2D convolutions consider both spatial and time dimensions, with each filter defining a different convolution volume. We perform the same computation on all the frames to reduce the number of parameters, transforming the events embedded within them to a different, more useful representation. This shared-events filters CNN was not converted here to a fully spiking implementation. In future work, it might be implemented as SNN by utilizing recurrent network topology. In this network design, each event frame should be separately processed in the first two spatial convolution layers. With SNN, an integrator, defined with a recurrent connection, can be utilized as a memory unit, comprising $H \times W \times (T * C)$ dimensions, collecting C temporal features. Three additional convolution layers will be applied to the integrator to process the whole Spatio-temporal data to predict the Laplacian.

We show that SNNs perform fairly well, despite featuring a slow maximal firing rate of only 100 Hz. Note that in contrast to its non-spiking version, inputs are sequentially presented to the network for only 100 mSec each. Along with the low-pass synapse filters that average input information over time, allow previous input data to persist in the network and interfere with the generation of a new predicted intensity image. A reset switch can be further implemented, with which reconstruction might be improved.

	#	filters	params	PSNR	SSIM	MSE
$\hat{L}$+PI	1	50,100,50,20,1	53,941	24.29	0.864	0.0042
	2	50,50,50,20,1	28,891	24.31	0.859	0.0042
	3	20,20,20,20,1	5,581	24.47	0.858	0.0042
	4	10,10,10,10,1	1,691	24.67	0.861	0.0039
	5	3,3,3,3,1	277	24.23	0.844	0.0042
	6	3,1,3,1,1	205	23.86	0.838	0.0047
	S_M	3*,2*,3,3,1	93	23.04	0.824	0.0058
	S_R	3*,2*,3,3,1	93	11.40	0.326	0.0734
	$SNN^{.1}$	3,3,3,3,1	277	17.90	0.679	0.0217
	SNN^5	3,3,3,3,1	277	19.36	0.756	0.0147
$\hat{I}$	1	50,100,50,20,1	53,941	19.53	0.519	0.0117
	4	10,10,10,10,1	1,691	19.60	0.512	0.0114
	5	3,3,3,3,1	277	19.22	0.490	0.0124

Table 1. Architecture and Performance table. $\hat{L} + PI$ stands for the neural networks, which feature Laplacian prediction followed by Poisson integration (reconstruction via Laplacian prediction). $\hat{I}$ stands for direct reconstruction. S_M and S_R stand for shared-event filters CNN, with Mish and ReLU activations. respectively. Filters marked with * were applied to each temporal channel (see text). $SNN^{.1}$ and SNN^5 stand for a full spiking implementation of model #5, with a maximal firing rate 100 Hz and 5 kHz, respectively. PSNR, SSIM and MSE metrics were calculated on the final reconstructed image.

Our models were demonstrated on the N-MNIST and N-Caltech101 datasets, which were "artificially" created with a saccade-moving event camera [21]. Data acquisition was based on a fixed non-biologically plausible three saccades. Our method is therefore limited to this precise method of data acquisition. However, we hypothesized that our Laplacian prediction- and Poisson integration-based methods could be utilized for arbitrary event data with relatively simple modifications. A recurrent architecture can be used without a U-shaped network in which skip connections are utilized to fill in pixel intensity within the image's gradients.

Acknowledgments

This work was supported by the Israel Innovation Authority (EzerTech) and the Open University of Israel research grant.

References

[1] G. Indiveri and R. Douglas, "Neuromorphic vision sensors," *Science,* vol. 288, no. 5469, pp. 1189-1190, 2000.

[2] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco and T. Delbruck, "Retinomorphic event-based vision sensors: bioinspired cameras with spiking output," *Proceedings of the IEEE,* vol. 102, no. 10, pp. 1470--1484, 2014.

[3] O. Ronneberger, P. Fischer and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, 2015.

[4] H. Rebecq, R. Ranftl, V. Koltun and D. Scaramuzza, "Events-to-video: Bringing modern computer vision to event cameras," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[5] H. Rebecq, R. Ranftl, V. Koltun and D. Scaramuzza, "High speed and high dynamic range video with an event camera," *IEEE transactions on pattern analysis and machine intelligence,* 2019.

[6] C. Scheerlinck, H. Rebecq, D. a. B. N. Gehrig, R. Mahony and D. Scaramuzza, "Fast image reconstruction with an event camera," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020.

[7] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks,* vol. 10, no. 9, pp. 1659--1671, 1997.

[8] C. Eliasmith and C. H. Anderson, Neural engineering: Computation, representation, and dynamics in neurobiological systems, MIT press, 2003.

[9] Y. Zaidel, A. Shalumov, A. Volinski, L. Supic and E. E. Tsur, "Neuromorphic NEF-based inverse kinematics and PID control," *Frontiers in Neurorobotics,* vol. 15, 2021.

[10] E. E. Tsur and M. Rivlin-Etzion, "Neuromorphic implementation of motion detection using oscillation interference," *Neurocomputing,* vol. 374, pp. 54--63, 2020.

[11] C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang and D. Rasmussen, "A large-scale model of the functioning brain," *Science,* vol. 338, no. 6111, pp. 1202--1205, 2012.

[12] T. a. B. J. Bekolay, 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.

[13] A. Hazan and E. Ezra Tsur, "Neuromorphic Analog Implementation of Neural Engineering Framework-Inspired Spiking Neuron for High-Dimensional Representation," *Frontiers in Neuroscience,* vol. 15, no. 109, 2021.

[14] K. Fischl, A. Andreou, T. Stewart and K. Fair, "Implementation of the neural engineering framework on the TrueNorth neurosynaptic system," in *IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2018.

[15] C.-K. Lin, A. Wild, G. Chinya, Y. Cao, M. Davies, D. M. Lavery and H. Wang, "Programming spiking neural networks on intel's loihi," *Computer,* vol. 51, no. 3, pp. 52-61, 2018.

[16] K. Boahen, "A neuromorph's prospectus," *Computing in Science & Engineering,* vol. 19, no. 2, pp. 14-28, 2017.

[17] A. Mundy, J. S. T. Knight and S. Furber, "An efficient SpiNNaker implementation of the neural engineering framework," in *International Joint Conference on Neural Networks (IJCNN)*, 2015.

[18] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: opportunities and challenges," *Frontiers in neuroscience,* vol. 12, p. 774, 2018.

[19] P. Perez, M. Gangnet and A. Blake, "Poisson image editing," in *ACM SIGGRAPH*, 2003.

[20] D. Misra, "Mish: A self regularized non-monotonic neural activation function," *arXiv preprint,* p. 1908.08681, 2019.

[21] 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.

[22] T. Simchony, R. Chellappa and M. Shao, "Direct Analytical Methods for Solving Poisson Equations in Computer Vision Problems," *IEEE transactions on pattern analysis and machine intelligence,* vol. 12, no. 5, pp. 435--446, 1990.

[23] H. Kim, A. Handa, R. Benosman, S.-H. Ieng and A. J. Davison, "Simultaneous mosaicing and tracking with an event camera," *J. Solid State Circ,* vol. 43, pp. 566-576, 2008.

[24] H. Kim, S. Leutenegger and A. J. Davison, "Real-time 3D reconstruction and 6-DoF tracking with an event camera," in *European Conference on Computer Vision*, 2016.

[25] H. Rebecq, T. Horstschafer, G. Gallego and D. Scaramuzza, "EVO: A Geometric Approach to Event-Based 6-DOF Parallel Tracking and Mapping in Real Time," *IEEE Robotics and Automation Letters,* vol. 2, no. 2, pp. 593-600, 2016.

[26] S. Barua, Y. Miyatani and A. Veeraraghavan, "Direct face detection and video reconstruction from event cameras," in *2016 IEEE winter conference on applications of computer vision (WACV)*, 2016.

[27] M. Mostafavi, L. Wang and K.-J. Yoon, "Learning to reconstruct hdr images from events, with applications to depth and flow prediction," *International Journal of Computer Vision,* pp. 1--21, 2021.

[28] B. su, L. Yu and W. Yang, "Event-Based High Frame-Rate Video Reconstruction With A Novel Cycle-Event Network," in *IEEE International Conference on Image Processing (ICIP)*, 2020.

[29] L. Wang, T.-K. Kim and K.-J. Yoon, "Eventsr: From asynchronous events to image reconstruction, restoration, and super-resolution via end-to-end adversarial learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[30] L. Mescheder, A. Geiger and S. Nowozin, "Which training methods for GANs do actually converge?," in *International conference on machine learning*, 2018.

[31] M. Mostafavi, J. Choi and K.-J. Yoon, "Learning to Super Resolve Intensity Images from Events," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[32] R. Massa, A. Marchisio, M. Martina and M. Shafique, "An efficient spiking neural network for recognizing gestures with a dvs camera on the loihi neuromorphic processor," *arXiv preprint,* p. 2006.09985, 2020.

[33] M. Osswald, S.-H. Ieng, R. Benosman and G. Indiveri, "A spiking neural network model of 3D perception for event-based neuromorphic stereo vision systems," *Scientific reports,* vol. 7, no. 1, pp. 1-12, 2017.

[34] Z. Jiang, Z. Bing, K. Huang and A. Knoll, "Retina-based pipe-like object tracking implemented through spiking neural network on a snake robot," *Frontiers in neurorobotics,* vol. 13, p. 29, 2019.

[35] S. Seifozzakerini, W.-Y. Yau, B. Zhao and K. Mao, "Event-Based Hough Transform in a Spiking Neural Network for Multiple Line Detection and Tracking Using a Dynamic Vision Sensor," in *BMVC*, 2016.

[36] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE transactions on image processing,* vol. 13, no. 4, pp. 600-612, 2004.

[37] C. Godard, O. Mac Aodha and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[38] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint,* p. 1803.08375, 2018.

[39] Keras, 2021. [Online]. Available: https://keras.io/. [Accessed 22 3 2021].

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint,* p. 1412.6980, 2014.

[41] D. Rasmussen, "NengoDL: Combining deep learning and neuromorphic modelling methods," *Neuroinformatics,* vol. 17, no. 4, pp. 611--628, 2019.

[42] V. A. Volpert, Elliptic Partial Differential Equations., Springer, 2011.