EDeNN: Event Decay Neural Networks for low latency vision

Celyn Walters
https://www.surrey.ac.uk/people/celyn-walters
Simon Hadfield
https://www.surrey.ac.uk/people/simon-hadfield
Centre for Vision, Speech and Signal Processing (CVSSP)
University of Surrey
Guildford, UK

Abstract

Despite the success of neural networks in computer vision tasks, digital 'neurons' are a very loose approximation of biological neurons. Today's learning approaches are designed to function on digital devices with digital data representations such as image frames. In contrast, biological vision systems are generally much more capable and efficient than state-of-the-art digital computer vision algorithms. Event cameras are an emerging sensor technology which imitates biological vision with asynchronously firing pixels, eschewing the concept of the image frame. To leverage modern learning techniques, many event-based algorithms are forced to accumulate events back to image frames, somewhat squandering the advantages of event cameras.

We follow the opposite paradigm and develop a new type of neural network which operates closer to the original event data stream. We demonstrate state-of-the-art performance in angular velocity regression and competitive optical flow estimation, while avoiding difficulties related to training Spiking Neural Networks. Furthermore, the processing latency of our proposed approach is less than $1/10$ any other implementation, while continuous inference increases this improvement by another order of magnitude. Code is available at https://gitlab.surrey.ac.uk/cw0071/edenn.

1 Introduction

Event cameras are a neurologically inspired visual sensor with a rising popularity in computer vision research. They are able to capture data with a much higher temporal resolution than traditional cameras, effectively overcoming motion blur. Their high dynamic range is also useful for detecting small changes and assists in low-light conditions. The reduced latency also makes it feasible to react rapidly to external stimulus [5]. As opposed to traditional frame-based cameras, individual events are streamed asynchronously. Unfortunately, since the majority of computer vision algorithms assume synchronous pixel measurements, it can be challenging to adapt these algorithms to work with the data stream produced by event cameras. For this reason, much existing work on event cameras has followed the naïve approach of accumulating the events into image frames at a fixed framerate [31]. In this paper, we argue that this aggregation neglects the main benefits of this kind of sensor modality. The sensor's true potential can only be realised by introducing new processing techniques which cater to the characteristics of event streams. To this end we propose EDeNN. Figure 1 contrasts our proposed approach against other standard network types.

Despite the fact that CNNs are conceptually based on neurological structures, both the sensory input and the perception are catered towards synchronous parallel hardware (cameras) and common data storage formats (images). With a view to treating event streams

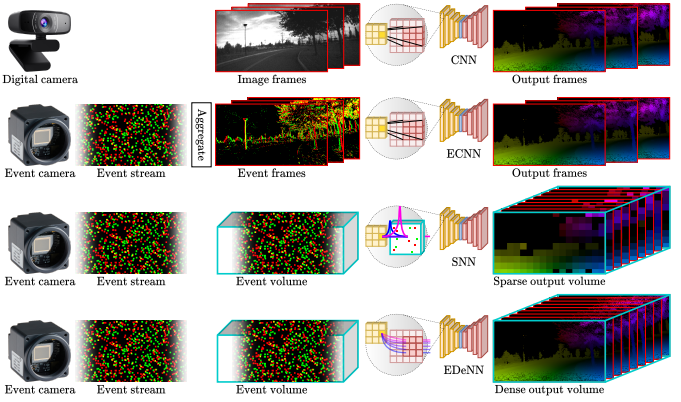
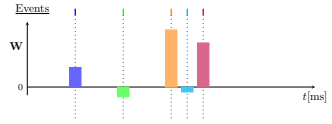


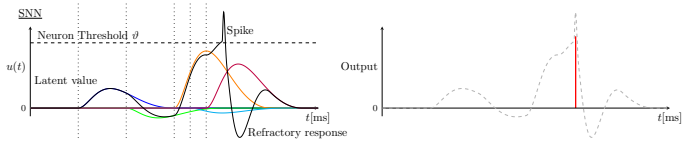
Figure 1: A comparison between neural networks. **Row 1**: Convolutional Neural Network (CNN) operates on image frames individually to produce output frames. **Row 2**: Event-based CNN ('ECNN') also operates on frames aggregated and quantised from the event stream. **Row 3**: Spiking Neural Network (SNN) operates on events directly and can produce output with high temporal resolution, but limited spatial resolution. **Row 4**: Proposed Event Decay Neural Network (EDeNN) also operates on events directly, but can produce output with both high temporal resolution and full spatial resolution.

neuromorphically, our proposed EDeNN approach also takes some inspiration from SNNs. SNNs are designed to more closely approximate biological neuron activity. In contrast to a CNN, each neuron in an SNN fires asynchronously and has an impulse response which only propagates discrete neural 'spikes' to subsequent neurons. The series of spikes reaching a given neuron is known as a spike train. Each spike in the train contributes to the neuron's membrane potential which is a continuously decaying state value. Once a neuron's membrane potential overcomes a certain threshold, that neuron transmits a spike to neurons in the next layer, followed by a short refractory period where it is less affected by additional incoming spikes. Figure 2b shows how individual events with positive and negative weightings affect the membrane potential and lead to an output spike train. Events generated from event cameras are already in the ideal data representation for SNNs, and an SNN layer produces another spike train as output.

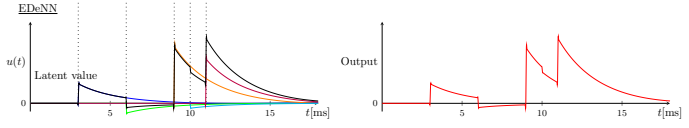
Although an SNN may be more suited to the data representation of event cameras, in practice it is far more difficult to train an SNN than a traditional CNN. This often limits applications to classification tasks [2]. A neuron can only transmit up to a single output spike for each incoming spike, usually much fewer. This means that every subsequent layer reduces the amount of information flowing through the network. There are clear similarities between this issue and the vanishing gradient problem in traditional CNNs. However, for event-driven learning, it is exacerbated by the discretisation of the spike train and the fact that neuron weightings cannot be updated when no output spikes occur. This significantly slows down the training speed of SNNs as large portions of the training data become unable to affect many of the network parameters. It also leads to wasted network capacity due to the 'dead neuron' problem: If a neuron's weightings accidentally reach values where they do not cause output spikes for any of the training samples, that neuron will never have its weights updated and will remain unresponsive forever. To mitigate this, the majority of SNN implementations seek to regress a small number of scalar values from full input images, such as angular velocity regression [5] and action categorisation [28]. By reducing the spatial dimensions in this way, the number of spikes in the output spike train can be maintained by pooling from neighbouring pixels. SNNs are historically poorly suited to learning dense estimation tasks such as optical flow estimation or semantic segmentation. To fully leverage the advantages of event cameras, our EDeNN approach is designed to take the best from both CNNs and SNNs in terms of accuracy and latency. In summary, we make the following contributions.



(a) Input spike train which with positive and negative weightings.



(b) SNN behaviour. **Left**: Black line shows the neuron's membrane potential, which has a lead-in time for each input spike. Output spikes are followed by a refractory response. **Right**: Red line shows an output spike when the membrane potential exceeds the neuron threshold.



(c) Behaviour of our proposed EDeNN model. **Left**: Black line shows the neuron's latent value **Right**: Red line shows that the continuous latent value comprises the output.

Figure 2: Dynamic responses from a single neuron given a weighted 1-dimensional spike train.

1) We define EDeNNs, a new approach to deep learning with event streams. Networks built with this paradigm may utilise events directly without accumulating and quantising into low-frequency 'event image' frames. EDeNNs also emphasise the latency of decision making and can be deployed in an extremely efficient online streaming mode.
2) To deal with the sparsity inherent in raw event streams, we propose a new formalisation of partial convolutions which takes learned biases into account. This yields an improvement of 23 % on the original implementation [13].
3) To avoid dead neurons and the vanishing gradients/spikes problem, we propose a new activation model which propagates the neuron's latent value directly without the need for discretisation.
4) We make public a new library, built on the pytorch auto-differentiation engine, which allows researchers to develop and train their own EDeNN solutions to different problems.

2 Related Work

Event cameras enable some applications where traditional frame-based cameras would fail, such as those requiring high speeds with rapid reflexes or in low-light conditions. One area which has seen significant interest is the control of unmanned aerial vehicles, where Sanket et al. enable dynamic obstacle avoidance on a quadrotor [23], and Dietsche et al. track electrical power lines [3].

Before the rise of neural-networks for event data, many approaches were gradient-based. Benosman et al. fit local planes to spatio-temporal event clouds [1]. An edge moving through space over time manifests as a surface in the 3D volume, commonly referred to as a Surface of Active Events (SAE). The coefficients for the planes fitted to this surface encode the edge motion, enabling the estimation of optical flow. Methods such as this one require a point neighbourhood which is not too small or large for stable plane-fitting.

Although event cameras are rising in popularity in the field of computer vision, the vast majority of event-based neural networks are architecturally similar to those based on

traditional RGB images. The events are accumulated into frames at a consistent framerate, discarding information. Gehrig et al. argue that the accurate timestamps of events are crucial, and without them the performance degrades significantly [5]. Nevertheless, CNNs act upon frames or volumes, so events must be accumulated in some way before a traditional deep learning tool focused on CNNs can be applied. This inherently involves a loss of information as multiple events are combined. Increasing the temporal resolution of event slices also increases their sparsity and leads to inefficient use of memory, storage, and computation. As a result, a trade-off is usually made to preserve information and maintain efficiency. Some CNN techniques choose to store the event counts for each pixel location [19], while others incorporate the relative event timestamps to try to preserve more temporal information [31]. Despite many of these papers citing the temporal advantages of this technology, it is rare for authors to measure the processing latency of their learning approaches. It is therefore not possible to ascertain if this dominates the advantages of the sensor itself, and therefore whether such approaches are useful in practice. In contrast EDeNNs focus heavily on latency and efficient inference, and we explicitly contrast this against previous methods in Section 4.

SNNs also do not require a lossy information aggregation step, rather they operate directly on the event stream with each layer producing a series of discrete and asynchronous output events. There are also various approaches to model neuron activity with differing levels of approximation. The Hodgkin-Huxley neuron [11] uses four differential equations to compute the membrane potential, and is considered too computationally complex to use for the large number of neurons comprising an SNN. The Leaky Integrate and Fire (LIF) [8] model is commonly used with SNNs thanks to its much simpler formulation, although it cannot capture all the dynamical behaviours of real neurons. The Spike Response Model (SRM) [7] model is similar to LIF, but additionally includes refractory responses to output spikes (see Figure 2b). One of the main challenges with implementing SNNs is enabling backpropagation through differentiable functions. There have been many individual approaches, each formalising the problem differently. Some overcome the challenge by first training an Artificial Neural Network (ANN) and subsequently converting it to an SNN [12, 15, 20]. This usually negatively impacts the accuracy, although measures are often introduced in an attempt to overcome this. Many others only backpropagate the membrane potential at certain times, ignoring the temporal dependency between spikes. Shrestha and Orchard released SLAYER, a CUDA-accelerated software framework to train SNNs by representing the derivative of spike functions (using the SRM model) by a probability density function [28]. Unfortunately, the fact that every input spike produces one or fewer output spikes leads to the vanishing gradient/spike and dead neuron problems. This usually restricts the application of SNNs to scalar classification and regression problems. In contrast, an EDeNNs avoids the spike discretization step, mitigating vanishing gradients and completely eliminating the dead neuron problem.

Outside of SNNs, there have been a number of attempts to develop event-driven variants of various specialist neural network architectures. Phased LSTM [18] introduced the input/output/forget gate structure of traditional LSTMs for detecting long-term relationships in event-data. Similarly [17] attempts to adapt existing sparse convolution approaches for application to event streams. More recently [24] attempts to apply graph-neural-networks to event data.

Outside the field of event-camera processing, there have been some other attempts to apply traditional CNN deep learning tools to non-image data. Notably, PointNet [21] and its successors [22, 26, 29] attempt to process point cloud data which, like event data, tends to be extremely sparse. Graph Neural Networks (GNNs) go a step further and eliminate the concept of spatial neighbourhoods in images entirely, replacing it with the concept of connectivity in order to process graph-based data [27, 30]. Unfortunately, because these techniques were not developed with event cameras in mind, they make no concessions to the efficiency nor latency of their inference. They generally must have access to the entire point cloud or graph at once before a prediction can be made. In contrast, our proposed EDeNNs are capable of extremely efficient low-latency streaming inference.

3 Methodology

Events originating from a neuromorphic sensor such as an event camera are streamed continuously using a sparse index style notation. This is generally formatted as $\mathcal{E} = [\boldsymbol{x}, p, t]$, with each event comprising image coordinates $\boldsymbol{x} = [x, y]$, polarity p and timestamp t. These sparse indices can be used to recreate an event volume $I \in \mathbb{R}^{W \times H \times C \times T}$ with similar spatial and temporal properties to those common in temporal CNNs.

$$I(\boldsymbol{x}, p, t) = \begin{cases} 1, & \text{if } [\boldsymbol{x}, p, t] \in \mathcal{E} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The building block of an EDeNN is the Event Decay Convolution (EDeC) layer. Inspired by SNNs, the EDeC layer also uses a decaying latent value to find temporal associations between sparsely distributed events. However, inspired by CNNs this is combined with a spatial convolution operation to recognise structural scene elements. Note that unlike an SNN, the EDeC layer does not discretise its output into a spike train. The latent value itself is propagated to the next layer, avoiding a decrease in information density leading to vanishing gradients/spikes and dead neurons.

Each EDeC layer is comprised of a number of EDeC neurons, which independently produce a single channel of the layer's output volume. In the most general sense, a single EDeC neuron performs a 3D spatio-temporal convolution of its input I and a learned spatio-temporal kernel $\hat{K} \in \mathbb{R}^{\hat{W} \times \hat{H} \times \hat{T}}$ (note that to help distinguish 2D and 3D convolutions, we indicate the domain of the convolutions using ':')

$$E(:, \bar{c}, :) = \sum_{c \in C} \hat{K}_c^{\bar{c}} * I(:, c, :). \tag{2}$$

However, in an EDeC neuron the kernel $\hat{K}$ is constrained to model only a certain class of functions. Specifically, each EDeC neuron is designed to exhibit the Markov property. This means that for each new slice in the input volume I, we only need the result for the previous slice and our new input slice. This makes it possible to perform streaming inference, where every incoming event slice is immediately processed and a prediction computed, without the need to reprocess the event volume.

In this paper, we enforce this by parameterising each EDeC neuron as a set of $K \in \mathbb{R}^{\hat{W} \times \hat{H}}$ spatial parameters plus one temporal decay parameter $\gamma \in [-1..1]$. As with SNNs, the decay rate is a parametric mathematical function. However, unlike SNNs we make the parameters of these decay functions learnable, and varying across neurons. We thus define the spatio-temporal kernel $\hat{K}$ as

$$\hat{K}_c(\boldsymbol{x}, t) = K_c(\boldsymbol{x}) \gamma^{\hat{T} - t}. \tag{3}$$

3.1 Filter separability for streaming inference

Thanks to this particular parameterisation, an EDeC neuron's spatio-temporal convolution filter $\hat{K}$ is linearly separable into two components which are highly effective for event based learning. These roughly translate to a spatial CNN element and a temporal SNN component. To see this, we first note that we can combine equations Equation (2) and Equation (3) to define the output of an EDeC neuron, as a weighted summation over previous timesteps

$$E(:, \bar{c}, t) = \sum_{c \in C} \sum_{\tau = t}^{t + \hat{T}} K_c^{\bar{c}} * I(:, c, \tau) \gamma^{\hat{T} - \tau}. \tag{4}$$

Next we move the final item $\tau = t + \hat{T}$ outside of the summation along with a power of gamma

$$E(:,\bar{c},t) = \sum_{c \in C} \left[K_c^{\bar{c}} * I(:,c,t) + \gamma \sum_{\tau=t}^{t+\hat{T}-1} K_c^{\bar{c}} * I(:,c,\tau) \gamma^{\hat{T}-\tau-1} \right]. \tag{5}$$

Finally, we note that the right hand side of the equation is equivalent to Equation (4) at $t-1$. This enables us to produce a definition which is recursive in time

$$E(:,\bar{c},t) = \sum_{c \in C} \left[K_c^{\bar{c}} * I(:,c,t) + \gamma E(:,c,t-1) \right]. \tag{6}$$

We can generalize this beyond the first layer of the network. Thanks to this separability, the forward pass of an EDeC neuron at any layer l and time slice t can be defined as the sum of two parts. First, the convolved output of the slice at the same time t in the previous layer $l-1$, and secondly the decayed output of the previous time slice $t-1$ at the current layer

$$E^l(:,\bar{c},t) = \sum_{c \in C} \left[K_c^{\bar{c}} * E^{l-1}(:,c,t) + \gamma E^l(:,c,t-1) \right]. \tag{7}$$

There are two main advantages of this separable formulation, which drastically improve the computational efficiency of an EDeNN. Firstly, both components are simple to compute. The 2D convolution operation has a number of parameters and complexity that scales with the square of the kernel size rather than its cube. Meanwhile, the second term can be pre-computed before the next time slice arrives. This helps maintain a low processing latency. However, the second and more fundamental implication of our separable formulation is the nature of the receptive field at deeper layers. The Markovian property of the temporal operation ensures that the output at every layer of the network will only depend on, at most, a single previous timestep. In contrast, for a standard spatio-temporal CNN convolution, the receptive field grows for each subsequent layer. This is problematic, as temporal expansion of the receptive field implies that the CNN must wait for future slices to be observed before it is able to make predictions about past slices. This greatly increases the latency of the prediction, and precludes true streaming inference with CNNs.

3.2 Weighted Partial Convolutions

Although our formulation supports efficient temporal propagation of events, the raw event tensor tends to be very sparse. As a result, the empty regions dominate the values produced by normal convolutions. More importantly these empty regions dominate the model's weight updates. *Partial* convolutions are more suited to these sparse tensors.

Partial convolutions have previously been presented for applications to image inpainting [13] and for zero-padding [14]. On the first layer of a network, a binary mask $\mathbf{M}$ is provided alongside the input tensor. This mask normally represents holes or zero-padding regions. The convolution operation is subsequently disabled for masked regions, and the unmasked pixels are convolved as if the holes were not present. This is shown to be beneficial in terms of computational efficiency and training stability. In the original formalisation by Liu et al., the elements of E^l are masked during convolution, and the resulting elements of E^{l+1} are multiplied by a scaling factor α based on the number of surrounding masked cells. This removes the effect of holes on the unmasked regions. We extend this idea to EDeC neurons as

$$E^l(:,\bar{c},t) = \alpha^l(:,t) \sum_{c \in C} \left[K_c^{\bar{c}} * E^{l\text{-}1}(:,c,t) \odot \mathbf{M}^{l\text{-}1}(:,t) \; + \gamma E^l(:,c,t\text{-}1) \odot \mathbf{M}^l(:,t\text{-}1) \right]$$

$$\alpha^l(\boldsymbol{x},t) = \frac{2|\Omega|}{\sum_{\delta \in \Omega} \left[\mathbf{M}^{l\text{-}1}(\boldsymbol{x}+\delta\boldsymbol{x},t) + \mathbf{M}^l(\boldsymbol{x}+\delta\boldsymbol{x},t\text{-}1) \right]}, \tag{8}$$

where $\odot$ represents the Hadamard product, α is the scaling factor and Ω is the domain of 2D offsets within the kernel K. In the case of a division by zero (*i.e.* the masks are empty) the value of α is set to zero.

At the first layer of the EDeNN, the mask is determined as the areas where events of either polarity occur in the input. For subsequent layers of the EDeNN, we define the mask using the scaling factor above.

$$\mathbf{M}^0(\boldsymbol{x},t) = I(\boldsymbol{x},1,t) + I(\boldsymbol{x},-1,t). \tag{9}$$

$$\mathbf{M}^l(\boldsymbol{x},t) = \begin{cases} 1, & \text{if } \alpha^l(\boldsymbol{x},t) > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

This approach closely mirrors that of the original partial convolution definition for CNNs in that it successfully ignores the presence of the masked cells by 'averaging out' their result. To be explicit, the value of E is invariant to the unmasking of spatial element $\bar{\mathbf{M}}^{l-1}$, under the following condition:

$$\sum_{c\in C} K_c^{\bar{c}} * E^{l-1}(:,c,t) \odot \bar{\mathbf{M}}^{l-1} = \frac{E^l(:,\bar{c},t)}{2|\Omega|}. \tag{11}$$

Similarly, the value of E remains unchanged when unmasking temporal element $\bar{\mathbf{M}}^l$ under the following condition:

$$\sum_{c\in C} \gamma E^l(:,c,t\text{-}1) \odot \bar{\mathbf{M}}^l = \frac{E^l(:,\bar{c},t)}{2|\Omega|}. \tag{12}$$

Thus we can imagine that the masking operation does not affect the output value, only if we assume that the *impact* of the unobserved elements on the output would be equal to the average impact of the observed elements.

Unfortunately, it is apparent that this definition conflates the effect of the masked inputs and their corresponding kernel values. If an 'important' (*i.e.* high weight) element of the kernel is masked out, this causes the same change to α as if an 'unimportant' (*i.e.* low weight) kernel element is masked out. In fact, the conditions above will only hold true when assuming that the value of the ignored kernel elements are exactly counteracted by the values of the missing input elements. In the extreme case of a kernel element with weight 0 being masked, the conditions above imply that we can only maintain invariance to the masking operating if the masked input item had an infinite value.

Inspired by this observation, we deviate from the original partial convolution formulation and specify an updated scaling factor $\hat{\alpha}$ which accounts for the learned kernel weights in the masked region where a is the total unmasked spatial kernel weight and b is the total unmasked temporal kernel weight

$$\hat{\alpha}^l(\boldsymbol{x},\bar{c},t) = \frac{\gamma|\Omega| + \sum_{c\in C}\sum_{\delta\in\Omega} K_c^{\bar{c}}(\delta\boldsymbol{x})}{[a+\gamma b]}, \tag{13}$$

$$a = \sum_{c\in C}\sum_{\delta\in\Omega} K_c^{\bar{c}}(\delta\boldsymbol{x})\mathbf{M}^{l-1}(\boldsymbol{x}+\delta\boldsymbol{x},t), \tag{14}$$

$$b = \sum_{c\in C}\sum_{\delta\in\Omega} \mathbf{M}^l(\boldsymbol{x}+\delta\boldsymbol{x},t\text{-}1). \tag{15}$$

Intuitively, the new scaling factor measures the maximum importance of all input elements (measured by their learned kernel weightings) divided by the total importance of the unmasked

input elements. As with the original scaling factor, we note that when all inputs are observed $\hat{\alpha}$ is simply 1 and when all inputs are unobserved we set $\hat{\alpha}$ to 0. However, we can see that under the proposed scheme, masking or unmasking an important (high weight) element would be expected to cause a larger change in the scaling factor. Meanwhile, (un)masking an unimportant (0 weight) item will no longer cause any change in the output, regardless of the value of corresponding input element.

We believe that these properties make our EDeC neuron a better choice for building layers which are invariant to input masking. Therefore we expect learned EDeNNs to generalise more successfully to variations in the masking of the input event volume (*i.e.* variations in the arrangements of sparse events).

4 Results

To show the generalisability of our EDeNN approach to event streams, we evaluate the latency as well as the accuracy in two different scenarios:

1. We estimate angular velocity for a moving camera, comparing directly with both an SNN and traditional neural networks, This is a traditionally good problem for SNNs because it is high frequency and the spatial aggregation helps prevent dead neurons. We also perform an ablation study for different kernel operators.
2. We perform dense estimation of optical flow, which is particularly challenging for SNNs. We compare against the current state of the art traditional CNNs for event-based optical flow estimation.

4.1 Regression with EDeNNs

Gehrig et al. present a dataset and approach [5] for angular velocity regression (i.e. estimating the rate of change of roll, pitch and yaw) using the SLAYER SNN framework [28]. The dataset simulates an event camera being shaken at different rates, with challenging saccadic motion. Although the regression problem itself is relatively low dimensional, the challenge comes from the high estimation rate required, and correspondingly low amount of information, which exacerbates vanishing gradients. Each sample is 100 ms in length at a spatial resolution of 240×180. Gehrig et al.'s approach was shown to be competitive with CNN baselines with the same number of layers. Notably, ResNet-50, a much deeper network gets the highest performance. Although they note that deeper SNNs *may* have better performance, it is also likely to be more challenging to train, due to the issues with vanishing gradients. In our evaluation we sought to keep as many aspects of the network architecture the same in our EDeNN, to allow for a more direct comparison. Our model has an equivalent configuration of layers (detailed in the supplementary material). It was trained for 500 epochs with batch size of 4, and we match the settling time protocol of [5] by evaluating the loss from 50 ms.

Approach	Data	Relative error	RMSE	Step time
ANN-6	V	0.22	59.00	–
ResNet-50 [10]	A	0.22	66.80	–
ResNet-50 [10]	V	0.15	36.80	–
SNN-6 [5]	E	0.26	66.32	0.15
EDeNN	E	**0.12**	**27.99**	**0.08**

Table 1: Comparison of angular velocity regression approaches. The data structures (E, A, V) are event-based [5], accumulation-based [16] and voxel-based [4], respectively. RMSE is in (°/s), step times are in (s).

As shown in Table 1, our approach is clearly more accurate than all other approaches. We have less than half of both the relative error and RMSE compared to the SNN. Indeed we are

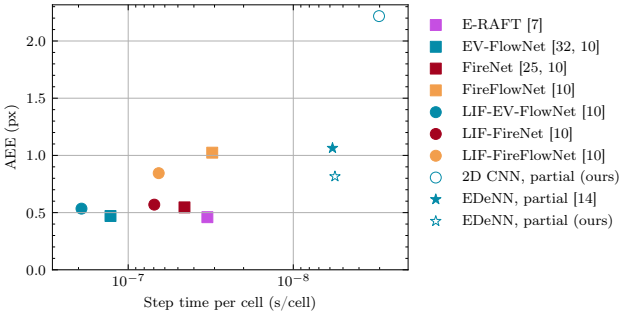


Figure 3: Comparison of approaches on the MVSEC [32] dataset. AEE is the Average Endpoint Error in pixels. Step time per cell is the average processing time divided by the number of elements in the input data. Square markers are CNNs, round are SNNs, and stars are EDeNNs.

even able to clearly outperform much deeper traditional CNNs such as ResNet-50. In addition to our increased accuracy, we also report a much lower average processing time over the test set than the state-of-the-art SNN. We should note that in order to provide the fairest possible comparison, this time was computed for a cold start over the entire 100 ms sample. In reality, an EDeNN, like an SNN, is able to stream new events at inference time. This means that once the network is primed, subsequent estimations could be performed at a fraction of this cost. Furthermore, this is all possible without modifying or pre-processing the event data through accumulation (A) or voxelisation (V).

4.2 Dense estimation with EDeNNs

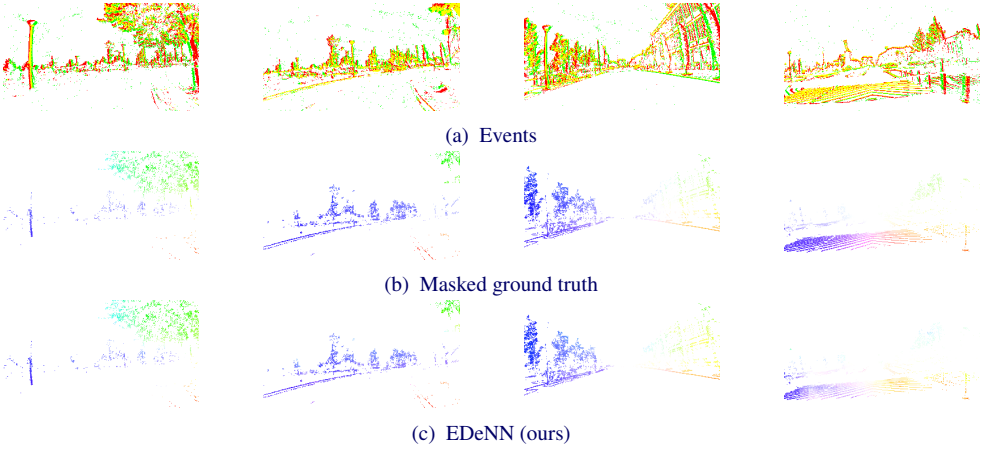
We next demonstrate the capabilities of EDeNNs for dense estimation tasks. These are commonplace with traditional CNNs but are particularly challenging for SNNs and implementations of this are extremely scarce.

For this task our EDeNN architecture is inspired by EV-FlowNet [31]. The overall architecture comprises four encoder and four decoder layers, and is arranged as a U-Net with concatenated residual connections. Intermediate optical flow estimates of differing resolutions are produced on the output of each decoder layer. Each of these intermediate flow estimations contributes to the overall loss function with equal weighting.

We use a window size of 48 ms with bins of 2 ms to correlate with the ground truth frame rate of 20 Hz, and evaluate the flow predictions of the final layer. By nature, event cameras do not produce events where there are no lighting or texture changes. The result of this is that it is almost impossible to estimate optical flow for subjects such as uniformly-coloured walls. Following the evaluation protocol of previous approaches, performance is computed after masking out pixels where there is no ground-truth flow or where there are no input events. We use a pixelwise L1 loss to train our EDeNN.

We estimate optical flow on the MVSEC dataset [32] and compare against many recent approaches. The results are shown quantitatively in Figure 3. The step times are measured on identical hardware and averaged over the sequence. We note that the SNN approaches [9] tend to have a slightly higher Average Endpoint Error (AEE) and a slightly worse step time per cell than their equivalent CNN counterparts. This is likely due to the vanishing gradient/spike issues which SNNs experience for dense prediction tasks and to an underexploitation of parallel computing architectures compared to the other techniques. Overall the top previous state-of-the-art approach appears to be E-RAFT [6] (evaluated on the original ground truth framerate/resolution) which achieves accuracy comparable to the EV-FlowNet baseline, but in

[0]Values taken from [5].



(a) Events

(b) Masked ground truth

(c) EDeNN (ours)

Figure 4: Results from the optical flow estimation task.

a step time which is 6 times lower. In contrast, our EDeNN approach, which has a similar layer structure to the original EV-FlowNet, is able to further decrease step time by a factor of 6 over E-RAFT. Our proposed EDeC neuron provides an error reduction of 63 % over using standard 2D convolutions ('2D CNN'), while our proposed reformulation of partial convolutions yields an error reduction of 23 % over the original formulation ('EDeNN partial [13]').

Qualitative results for the EDeNN are shown in Figure 4. We can see that generally speaking the orientation of the estimated flows (shown as hue) match those of the ground truth. The hue difference on the fourth row of Figure 4 is located on the road, an image region which does not usually contain events in this dataset. It is also very interesting to note that our EDeNN flow network performs very well at complex structures with fine details such as tree foliage, which is an area where traditional optical flow techniques struggle. This suggests there is likely a strong synergy between traditional image based CNNs and EDeNNs.

5 Conclusions

This work proposes a new kind of neural network called Event Decay Neural Networks to overcome limitations of both CNNs and SNNs for event-based data. Because EDeNNs operate closer to the original event data stream, event accumulation is not necessary which preserves the high temporal resolution and lack of motion blur. Also, EDeNNs avoid the dead neurons and vanishing gradients/spikes problem of SNNs, assisting training for full-sized image outputs. This is possible by using a new Event Decay Convolution neuron which propagates continuous decaying latent values. We also propose a new formalisation of partial convolutions which caters to sparse event data by accounting for learned biases.

We showed that EDeNNs outperform the SNN and CNN baselines for angular velocity regression for both accuracy and step time. We also achieve competitive accuracy on optical flow estimation at full resolution, but with orders of magnitude reduction in processing times. The Markovian nature of EDeC kernels suggests that minimal calculation is required for subsequent events, enabling streaming inference and vastly reducing the practical latency/ computational complexity resources on real hardware.

Acknowledgements

This work was funded by the EPSRC under grant agreement EP/S035761/1.

References

[1] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi. Event-Based Visual Flow. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2):407–417, feb 2014. ISSN 2162-237X. doi: 10.1109/TNNLS.2013.2273537.

[2] Kenneth Chaney, Artemis Panagopoulou, and Kostas Daniilidis. Self-Supervised Optical Flow with Spiking Neural Networks and Event Based Cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[3] Alexander Dietsche, Giovanni Cioffi, Javier Hidalgo-Carrio, and Davide Scaramuzza. Powerline Tracking with Event Cameras. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6990–6997, 2021. ISBN 9781665417143. doi: 10.1109/iros51168.2021.9636824.

[4] Daniel Gehrig, Antonio Loquercio, Konstantinos Derpanis, and Davide Scaramuzza. End-to-end learning of representations for asynchronous event-based data. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-October(ICCV):5632–5642, 2019. ISSN 15505499. doi: 10.1109/ICCV.2019.00573.

[5] Mathias Gehrig, Sumit Bam Shrestha, Daniel Mouritzen, and Davide Scaramuzza. Event-based angular velocity regression with spiking networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[6] Mathias Gehrig, Mario Millhäusler, Daniel Gehrig, and Davide Scaramuzza. Dense Optical Flow from Event Cameras. In *IEEE Int. Conf. 3D Vis.(3DV)*, 2021.

[7] Wulfram Gerstner. Time structure of the activity in neural network models. *Physical Review E*, 51(1):738–758, jan 1995. ISSN 1063-651X. doi: 10.1103/PhysRevE.51.738.

[8] Wulfram Gerstner and Werner M. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002. doi: 10.1017/CBO9780511815706.

[9] Jesse Hagenaars, Federico Paredes-Vallés, and Guido de Croon. Self-Supervised Learning of Event-Based Optical Flow with Spiking Neural Networks. *Advances in Neural Information Processing Systems*, 34(NeurIPS), 2021.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:770–778, 2016. ISSN 10636919. doi: 10.1109/CVPR.2016.90.

[11] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544, aug 1952. ISSN 0022-3751. doi: 10.1113/jphysiol.1952.sp004764.

[12] Eric Hunsberger and Chris Eliasmith. Spiking Deep Networks with LIF Neurons. *arxiv*, pages 1–9, 2015.

[13] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting Chun Wang, Andrew Tao, and Bryan Catanzaro. Image Inpainting for Irregular Holes Using Partial Convolutions. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11215 LNCS:89–105, 2018. ISSN 16113349. doi: 10.1007/978-3-030-01252-6_6.

[14] Guilin Liu, Kevin J. Shih, Ting-Chun Wang, Fitsum A. Reda, Karan Sapra, Zhiding Yu, Andrew Tao, and Bryan Catanzaro. Partial Convolution based Padding. *ArXiV*, 2018.

[15] Qian Liu, Yunhua Chen, and Steve Furber. Noisy Softplus: an activation function that enables SNNs to be trained as ANNs. *arxiv*, 2017.

[16] Ana I. Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso Garcia, and Davide Scaramuzza. Event-Based Vision Meets Deep Learning on Steering Prediction for Self-Driving Cars. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (Dl):5419–5427, 2018. ISSN 10636919. doi: 10.1109/CVPR.2018.00568.

[17] Nico Messikommer, Daniel Gehrig, Antonio Loquercio, and Davide Scaramuzza. Event-based asynchronous sparse convolutional networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, pages 415–431. Springer, 2020.

[18] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. *Advances in neural information processing systems*, 29, 2016.

[19] Anh Nguyen, Thanh Toan Do, Darwin G. Caldwell, and Nikos G. Tsagarakis. Real-time 6DOF pose relocalization for event cameras with stacked spatial LSTM networks. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2019-June:1638–1645, 2019. ISSN 21607516. doi: 10.1109/CVPRW.2019.00207.

[20] Peter O'Connor, Daniel Neil, Shih Chii Liu, Tobi Delbruck, and Michael Pfeiffer. Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in Neuroscience*, 7(7 OCT):1–13, 2013. ISSN 16624548. doi: 10.3389/fnins.2013.00178.

[21] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua: 77–85, dec 2016. doi: 10.1109/CVPR.2017.16.

[22] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *Advances in Neural Information Processing Systems*, 2017-Decem:5100–5109, jun 2017. ISSN 10495258.

[23] Nitin J. Sanket, Chethan M. Parameshwara, Chahat Deep Singh, Ashwin V. Kuruttukulam, Cornelia Fermüller, Davide Scaramuzza, and Yiannis Aloimonos. EVDodge: Embodied AI For High-Speed Dodging On A Quadrotor Using Event Cameras. *arXiv*, 2019.

[24] Simon Schaefer, Daniel Gehrig, and Davide Scaramuzza. Aegnn: Asynchronous event-based graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12371–12381, 2022.

[25] Cedric Scheerlinck, Henri Rebecq, Daniel Gehrig, Nick Barnes, Robert E. Mahony, and Davide Scaramuzza. Fast image reconstruction with an event camera. *Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020*, pages 156–163, 2020. doi: 10.1109/WACV45572.2020.9093366.

[26] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D object proposal generation and detection from point cloud. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:770–779, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.00086.

[27] Weijing Shi and Ragunathan Rajkumar. Point-GNN: Graph neural network for 3D object detection in a point cloud. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1708–1716, 2020. ISSN 10636919. doi: 10.1109/CVPR42600.2020.00178.

[28] Sumit Bam Shrestha and Garrick Orchard. SLAYER: Spike Layer Error Reassignment in Time. In *Advances in Neural Information Processing Systems (NeurIPS)*, number NeurIPS, pages 1419—-1428, 2018.

[29] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep Convolutional Networks on 3D Point Clouds. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019-June:9613–9622, nov 2018. ISSN 10636919. doi: 10.1109/CVPR.2019.00985.

[30] Yanan Zhang, Di Huang, and Yunhong Wang. PC-RGNN: Point Cloud Completion and Graph Neural Network for 3D Object Detection. In *Proceedings of the AAAI conference on artificial intelligence*, 2020.

[31] Alex Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. EV-FlowNet: Self-Supervised Optical Flow Estimation for Event-based Cameras. In *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, jun 2018. ISBN 978-0-9923747-4-7. doi: 10.15607/RSS.2018.XIV.062.

[32] Alex Zihao Zhu, Dinesh Thakur, Tolga Ozaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The Multivehicle Stereo Event Camera Dataset: An Event Camera Dataset for 3D Perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039, jul 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2800793.