Real-time Classification from Short Event-Camera Streams using Input-filtering Neural ODEs

Abstract

Event-based cameras are novel, efficient sensors inspired by the human vision system, generating an asynchronous, pixel-wise stream of data. Learning from such data is generally performed through event integration into images. This requires buffering long sequences and can limit the response time of the inference system. In this work, we propose to directly use events from a DVS camera, which produces a stream of intensity changes and their spatial coordinates. This sequence is used as an input for a novel asynchronous RNN-like architecture, the Input-filtering Neural ODE (INODE). INODE allows for input signals to be continuously fed to the network, as done for filtering dynamical systems. INODE learns to discriminate short event sequences and to perform event-by-event online inference. We demonstrate our approach on a series of classification tasks, comparing against a set of LSTM baselines. We show that, independently of the camera resolution, INODE can outperform the baselines by a large margin on the ASL task and it is on par with a considerably larger LSTM for the NCALTECH task. Finally, we show that INODE is accurate even when provided with very few events.

1 Introduction

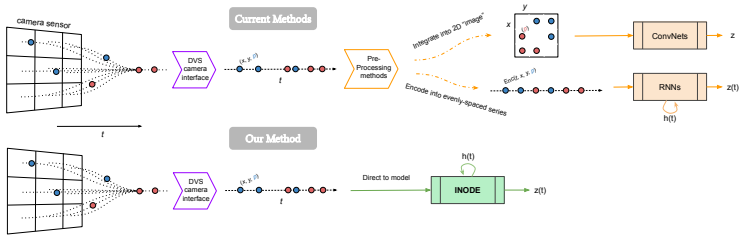


Figure 1: Approach rationale. The standard way to perform machine learning on asynchronous event stream data from DVS cameras consists in either integrating events into a 2D grid to be fed into convolutional networks or converting them into traditional time series with some time discretization scheme. On the contrary, our method requires no preprocessing or loss of information, and inherently handles the data stream's asynchronous timing. In the figure, blue and red dots represent events of different polarity respectively.

Event-based cameras are asynchronous sensors that capture changes in pixel intensity as binary events, with very high frequency compared to RGB sensors. This makes them suitable for high speed applications, such as robotics [16, 10] and other safety-critical scenarios. The Dynamic Vision Sensor (DVS) [20] is an event camera that, compared to traditional sensors, has low power consumption, high dynamic range, no motion blur, and microsecond latency times. Due to their asynchronous and binary format, there is no obvious choice of a model class for handling DVS data, unlike the predominant use of convolution-based models for RGB images. In this paper, we propose the use of a deep-learning and differential-equation hybrid method for such tasks, inspired by Neural Ordinary Differential Equations (NODE). NODE pioneered a novel machine learning approach where the data is modeled as an ODE in latent space, which can in principle be adjusted to process multiple asynchronous inputs. [9]. Most recent works using machine-learning to model DVS data integrate individual events to convert them into formats that can be fed as input into existing models, but lose precise timing information. The work of [1] studies the benefit of using precise temporal event data over aggregated event techniques. In particular, the study states:

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

The use of information theory to characterize separability between classes for each temporal resolution shows that high temporal acquisition provides up to 70% more information than conventional spikes generated from frame-based acquisition as used in standard artificial vision, thus drastically increasing the separability between classes of objects. This provides motivation to research methods that can directly handle asynchronous data.

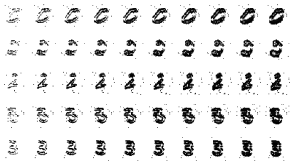


Figure 2: Event integration. This figure shows the result of integrating 100×10 consecutive events into a pixel grid. Our method trains and performs inference without directly integrating events, but instead processing one event at a time.

Summary of contributions. This work develops a novel real-time online classification model for event-based camera data streams. Moreover, it proposes INODE, an extension of the NODE architecture, which can directly take as input the stream of a possibly-high-frequency signal. This can be seen a continuous-time extension of Recurrent Neural Networks (RNNs). INODE is trained to perform continuous-time event filtering in order to infer classification labels online, based on its hidden state at a given moment. At test time, the classification prediction and the hidden state are updated as each (asynchronous) camera event is received. The event polarity and spatial coordinates are fed directly as inputs to the network without using convolutional layers or event integration. Importantly, we remark that input data is not processed in any form beyond normalization.

Summary of experiments. We demonstrate that the proposed approach excels in sample efficiency and real-time performance, significantly outperforming several LSTM architectures using short sequencing during online inference at test time. Furthermore, our method works with raw, noisy camera readings and is also invariant to the camera resolution used to capture the data.

2 Input-filtering Neural ODE

The proposed approach builds upon the architecture proposed in [25], with the difference that here we do not focus on the improvement of training efficiency and use standard back-propagation through time. We implement a batch Euler ODE solver so that our network can be dealt with as an RNN. This allows for the state to be unmeasured (hidden), for instance like in LSTMs. The result is a recurrent architecture with skip connections that can handle unevenly-spaced points in time. We also add a decoder network as a classifier.

Input-filtering Neural ODE. Consider the constrained differential optimization problem:

$$\begin{aligned} &\min_{\theta_f, \theta_g \in \mathbb{R}^m} \int_{t_0}^{t_1} L(z(t), \bar{z}(t))\, dt, \\ &\text{s.t.} \quad h'(t) = f(h(t), u(t); \theta_f), \quad z(t) = g(h(t); \theta_g), \quad h(t_0) = h_0, \end{aligned} \tag{1}$$

where $h(t)$ is the hidden state, $u(t)$ is the input, $z(t)$ is the predicted output, $\bar{z}(t)$ is the desired output, the loss L is given, f and g are neural networks with a fixed architecture defined by, respectively, θ_f, and θ_g which are parameters that have to be learned. The first two equality constraints in (1) define an ODE. Problems of this form have been used to represent several inverse problems, for instance in machine learning, estimation, filtering and optimal control [30, 17, 27]. Since this architecture can act as a general filter for the input signal, $u(t)$, we refer to it as the *Input-filtering Neural ODE* (INODE). We consider this a general framework for handling event data in a machine-learning scenario.

Application to DVS cameras. We propose to use INODE to build a system that predicts (labels) online by filtering a live-stream of DVS-camera events. The aim is to learn the ODE in problem (1), given short excitation event sequences $u(t)$. Ideally, this model should produce the fastest trajectory from the initial state h_0 to an appropriate (unknown) state $\bar{h}$ such that $\bar{z} = g(\bar{h})$, where g serves as a classification layer and $\bar{z}$ are the labels to be predicted. Hence, we fix the target to $\bar{z}(t) = \bar{z},\ \forall t$.

Event inputs. Events are high-frequency signals, and solving a high-frequency ODE is difficult. Event streams are also extremely dense: the time between events is, in general, very small (often $< 100\mu$s). We propose the use of a sample-and-hold approach, where events are held constant for up to a maximum delta-time $\mathrm{d}_{\max}$. In the rare case that no events occur after $\mathrm{d}_{\max}$, then we simply wait for the next event and hold the previous result without running the forward pass.

Problem discretization. A neuromorphic dataset D is a collection $\mathbf{e} = \{e_i\}_{i=0}^{M}$ of events $e_i = (x_i, y_i, p_i, t_i)$, where M is the number of events considered for a given sample (typically on the

order of thousands), and labels $\bar{z} \in \{0, ..., C-1\}$ for C classes. A digit is represented by a tuple $(\mathbf{e}, \bar{z})$ and the dataset by $D = \{(\mathbf{e}, \bar{z})_n\}_{n=0}^{N}$, where N is the number of samples. Thus, the integral in (1) is discretized for each sample using a subset of size S evaluation points $[t_1, ..., t_S]$ as: $\mathcal{L}(\mathbf{e}, \bar{z}) = \frac{1}{S}\sum_{i=1}^{S} L(z(t_i), \bar{z})$, where L is the cross-entropy loss. For each evaluation point, a new input event is used, i.e., $u(t_i) = (x(t_i), y(t_i), p(t_i))$. Finally, the sample loss is averaged over the dataset $\mathcal{L}_D = \mathbb{E}_{(\mathbf{e},\bar{z})}\left[\mathcal{L}(\mathbf{e}, \bar{z})\right]$ and used for optimization.

Time step normalization. To accurately use the time-steps dt, they can be normalized to values smaller than one (timestamps are recorded in microseconds and thus quickly reach very large values). At the same time, dt should not be very small to avoid optimization issues, such as vanishing gradients. We compute dt from the raw time-steps and divide by the 98th quantile d_q from the empirical distribution of dt for each training dataset, pre-computed and fixed, with an upper threshold at 1. The normalized step is $d\tau = dt/d_q$. The complete training procedure is summarised in Algorithm 1 (Appendix A).

3 Experiments

We consider multiple classification tasks to validate our method, benchmarking against LSTM variants. We always learn from short event subsequences (up to 100 events). Performance is evaluated with the same number of events used during training. This allows for potential real-time classification (when properly optimized), as inference time increases with number of events processed. We report full Tables and Figures for the experiment in the Appendix.

Setup. We use the same configurations, architectures, and hyper-parameters for all of the datasets and model variants. We train all models with different $\rho = \{1, 0.4, 0.2\}$ levels, where ρ is the fraction of train dataset used for training. For each sequence, we sample a random offset and relative sub-sequence of length $S \ll M$. In all of the experiments we set $S = 100$. We then use such sub-sequence as input $u(t)$ for the model with batch size $B_\rho = \rho B_{\rho=1}$. At test time, we consider different scenarios: a standard case, where the models are evaluated with $S = 100$ on the test set, and more challenging ones, in which they are evaluated with short sub-sequences in the range $S = \{10, 20, 30, ..., 100\}$.

Baselines. We first compare INODE against LSTM and bidirectional LSTM (bi-LSTM). The LSTMs and bi-LSTMs receive the event time-step as additional input. We consider three bi-LSTM models with hidden states of dimension $\{36, 72, 128\}$. The bi-LSTM$_{72}$ has approximately the same capacity of INODE, while bi-LSTM$_{128}$ is 3x larger. We also consider a variant of LSTM, the PhasedLSTM [22] without coordinate-grid embedding. This model explicitly handles asynchronous data learning an additional phase gate. Such approach is – according to the authors – fruitful for *long* sequences (>1000 steps), in which the phase gate can exploit periodic mechanism in the data. Given our use case, *short* sequences of events (<100), we do not expect improvements over a standard LSTM. To the best of our knowledge, this is the only known method which – like ours – inherently handles asynchronous timing within the model and does not need to learn an external transition model. Unfortunately, our initial results with standard PhasedLSTM were rather poor. However, combining phased and bidirectional LSTM seemed promising. We denote this as P-bi-LSTM.

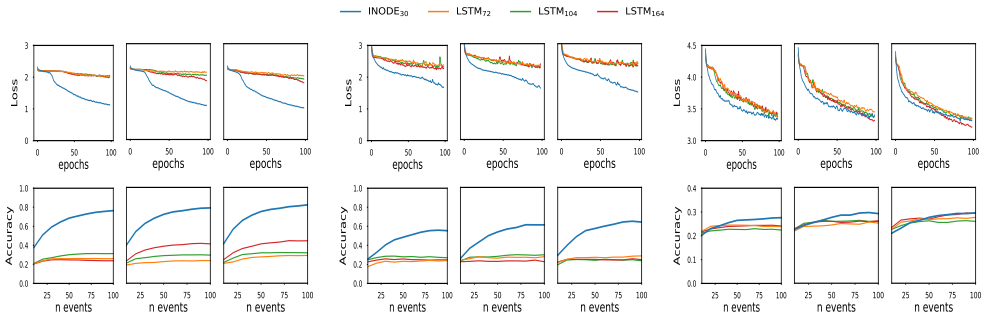
Datasets. We consider three neuromorphic datasets: **i) NMNIST** The NMNIST dataset [23] is a neuromorphic version of MNIST. It is an artificial dataset, generated by moving a DVS sensor in front of an LCD monitor displaying static images. It consists of 60k training samples and 10k test samples, for 10 different digits on a grid of 34 $\times$ 34 pixels. We consider only the first 2,000 (of potentially up to 6,000) events for each sequence. We do not stabilize the events spatially nor attempt to remove noisy events, which are options available in the dataset. **ii) ASL (12-16k)** The ASL-DVS dataset, is a neuromorphic dataset, obtained for a stream of real-world events [33]. It consists of around 100k samples for 24 different letters from the American Sign Language, with spatial resolution 180 $\times$ 240. Its sequences range from 1-500k events, with length distribution peaking in the 12-16k range. To avoid inconsistencies, we consider a subset containing only samples with a number of events between 12k and 16k. The resulting dataset contains 12,275 training samples plus 1,364 test samples. **iii) NCALTECH** The NCALTECH dataset [23] is the neuromorphic version of CALTECH101, produced in the same fashion as NMNIST. It consists of 100 heavily unbalanced classes of objects plus a background, with spatial resolution 172 $\times$ 232. The dataset contains 6,634 training samples and 1,608 test samples, after removing the background images. As with NMNIST, we again avoid stabilizing/denoising the images.

Solver. We train each model using ADAM for 300 epochs, with $S = 100$ and learning rate of 1e-3. The batch size $B_{\rho=1}$ is 1000 for NMNIST, and 100 for the other datasets. We consider a simple multi-layer perceptron for f: $f(x, u) = \text{FC}_3(\sigma(\text{FC}_2(\sigma(\{\text{FC}_1(x), \text{FC}_u(u)\}))$, where $\{\cdot, \cdot\}$ denotes the concatenation operation, FC is a fully-connected layer, and $\sigma = \tanh$ is the activation.

Results. When testing the models, we vary both the size of the *training* dataset and the number of *test* events used for the classification ($10 \leq S \leq 100$). The former is used to show INODE's learning efficiency when using a small amount of training data, while the latter demonstrates INODE's real-time scenario usability. Tables 4, 5, and 6 in Appendix C report accuracies for each of our datasets. The LSTM with 164 states outperforms the proposed architecture on NMNIST, see Table 6. On the ASL dataset (Table 5) our approach consistently outperforms all of the unidirectional baselines with a margin of 20%. We believe this is important since, among the considered datasets, ASL contains by far the most realistic data, being the only one not generated from static images. For NCALTECH, our approach is either on par or better than the LSTM when a small percentage of event is used (Table 4). For the bidirectional baselines, with approximately the same capacity (INODE$_{30}$ and bi-LSTM$_{72}$), INODE performs better then the bi-LSTMs on all of the datasets. Increasing the baseline capacity (bi-LSTM$_{128}$), INODE performs better on NCALTECH and ASL, while slightly losing its edge to the bi-LSTM$_{128}$ on NMNIST. Decreasing the training-set size has essentially no impact on NMNIST for all models – confirmation of a relatively simple dataset. One can also notice that, save a couple of exceptions on NMNIST, INODE outperforms the bidirectional methods regardless of number of input events. These are as low as $S = 10$, and, in principle, even $S = 1$ is possible without modifying our approach. Interestingly, with a mere 10 events, the model can correctly classify NMNIST digits about half of the time. As such, we demonstrate INODE's ability to extract information in the case of exceptional sparsity and data unavailability. This could be extremely important in scenarios such as collision avoidance and human-machine interaction, where safety is a paramount requisite. Finally, Figure 4, 5 and more comprehensive figures found in the Appendix further illustrate how INODE trains faster using fewer samples and events, especially on the ASL dataset.

Table 1: Classification accuracy on test sets for different datasets between INODE and comparable baselines. More baselines and results in Appendix C.

MODEL	DATASETS		
	NMNIST	ASL	NCALTECH
INODE$_{30}$	0.89	0.79	0.34
BI-LSTM$_{72}$	0.84	0.61	0.30
LSTM$_{72}$	0.81	0.35	0.31



(a) MNIST (100 epochs) (b) ASL (100 epochs) (c) NCALTECH (100 epochs)

Figure 3: Summary of results. Train/test losses and classification performance for INODE and multiple LSTM baselines, with increasing number of inference events per digit from 10 to 100. The three images for each dataset sub-figure correspond to training-set fraction of 20% (left), 40% (center), and 100% (right).

4 Conclusion

This paper presents a novel approach for performing machine learning from event-camera streams. The proposed INODE model is devised to handle high-frequency event data, inherently making use of the precise timing information of each individual event, and does not require processing the raw data into different formats. INODE excels in the most realistic scenarios, when little training data and few events are available. This makes it suitable for real-time, low-computation settings where decisions must be taken with only few event such as collision avoidance and high-speed object recognition.

References

[1] Himanshu Akolkar, Cedric Meyer, Xavier Clady, Olivier Marre, Chiara Bartolozzi, Stefano Panzeri, and Ryad Benosman. What can neuromorphic event-driven precise timing add to spike-based pattern recognition? *Neural computation*, 27:561–593, 03 2015.

[2] Himanshu Akolkar, Stefano Panzeri, and Chiara Bartolozzi. Spike time based unsupervised learning of receptive fields for event-driven vision. *Proceedings - IEEE International Conference on Robotics and Automation*, 2015:4258–4264, 06 2015.

[3] Patrick Bardow, Andrew J Davison, and Stefan Leutenegger. Simultaneous optical flow and intensity estimation from an event camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 884–892, 2016.

[4] 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)*, pages 1–9, March 2016.

[5] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[6] J.C. Butcher and G. Wanner. Runge-Kutta methods: some historical notes. *Applied Numerical Mathematics*, 22(1-3):113–151, November 1996.

[7] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. Asynchronous Convolutional Networks for Object Detection in Neuromorphic Cameras. *arXiv:1805.07931 [cs]*, May 2018. arXiv: 1805.07931.

[8] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. Matrix-LSTM: a Differentiable Recurrent Surface for Asynchronous Event-Based Data. *arXiv e-prints*, page arXiv:2001.03455, Jan 2020.

[9] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural Ordinary Differential Equations. In *NeurIPS*, 2018.

[10] Rika Sugimoto Dimitrova, Mathias Gehrig, Dario Brescianini, and Davide Scaramuzza. Towards low-latency high-bandwidth control of quadrotors using event cameras. *arXiv preprint arXiv:1911.04553*, 2019.

[11] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[12] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[13] Daniel Gehrig, Antonio Loquercio, Konstantinos G. Derpanis, and Davide Scaramuzza. End-to-End Learning of Representations for Asynchronous Event-Based Data. *arXiv:1904.08245 [cs]*, April 2019. arXiv: 1904.08245.

[14] Amir Gholami, Kurt Keutzer, and George Biros. ANODE: Unconditionally Accurate Memory-Efficient Gradients for Neural ODEs. Preprint at arXiv:1902.10298, 2019.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[16] Hanme Kim, Stefan Leutenegger, and Andrew J Davison. Real-time 3d reconstruction and 6-dof tracking with an event camera. In *European Conference on Computer Vision*, pages 349–364. Springer, 2016.

[17] Kody Law, Andrew Stuart, and Kostas Zygalakis. Data assimilation. *Cham, Switzerland: Springer*, 2015.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10:508, 2016.

[20] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128$\times$ 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, Feb 2008.

[21] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

[22] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased LSTM: Accelerating Recurrent Network Training for Long or Event-based Sequences. *arXiv:1610.09513 [cs]*, October 2016. arXiv: 1610.09513.

[23] Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 9:437, 2015.

[24] Lukas Paulun, Anne Wendt, and Nikola Kasabov. A retinotopic spiking neural network system for accurate recognition of moving objects using neucube and dynamic vision sensors. *Frontiers in Computational Neuroscience*, 12:42, 2018.

[25] Alessio Quaglino, Marco Gallieri, Jonathan Masci, and Jan Koutník. Snode: Spectral discretization of neural odes for system identification. In *International Conference on Learning Representations*, 2020.

[26] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. Events-to-video: Bringing modern computer vision to event cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3857–3866, 2019.

[27] I. Michael Ross. *A primer on Pontryagin's principle in optimal control*, volume 2. Collegiate publishers San Francisco, 2009.

[28] Yulia Rubanova, Ricky T. Q. Chen, and David Duvenaud. Latent ODEs for Irregularly-Sampled Time Series. *arXiv:1907.03907 [cs, stat]*, July 2019. arXiv: 1907.03907.

[29] Yusuke Sekikawa, Kosuke Hara, and Hideo Saito. Eventnet: Asynchronous recursive event processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2019.

[30] Robert F. Stengel. *Optimal control and estimation*. Dover books on advanced mathematics. Dover Publications, New York, 1994.

[31] Evangelos Stromatias, Miguel Soto, Teresa Serrano-Gotarredona, and Bernabé Linares-Barranco. An event-driven classifier for spiking neural networks fed with synthetic or dynamic vision sensor data. *Frontiers in neuroscience*, 11:350, 2017.

[32] Stepan Tulyakov, Francois Fleuret, Martin Kiefel, Peter Gehler, and Michael Hirsch. Learning an event sequence embedding for dense event-based deep stereo. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1527–1537, 2019.

[33] Alhabib Abbas Eirina Bourtsoulatze Yin Bi, Aaron Chadha and Yiannis Andreopoulos. Graph-based object classification for neuromorphic vision sensing. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2019.

[34] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. EV-FlowNet: Self-Supervised Optical Flow Estimation for Event-based Cameras. *Robotics: Science and Systems XIV*, June 2018. arXiv: 1802.06898.

Appendix

A Related Works

We review previous works related to our method, first describing alternative approaches to process events and discussing their relative advantages, then briefly introducing NODE methods.

Learning from event data. Event data from DVS cameras, being asynchronously streamed per sensor array pixel, requires careful processing to be compatible with traditional machine learning models. Methods for handling event data can be, in general, divided into grouped-event-based and per-event-based. The former employ a scheme to integrate multiple events into a single data structure that can be handled by spatially-based (e.g., convolutional) models, while the latter process the data stream on an event-by-event basis. Figure 1 illustrates the main differences between the reviewed works and the proposed approach.

Grouped-event methods. One of the more evident strategies in this category is to integrate time windows of data into grayscale intensity images, then apply existing computer vision techniques on these reconstructions. This is used, for example, in optical flow estimation [3], SLAM [16] and face recognition [4]. Such a process requires various filtering, tracking, and/or inertial measurement integration to properly compute frame offsets. This integration method itself is also the subject of [26], that uses RNNs to obtain usable intensity video from events. The main advantage of these methods is the possibility of directly plugging-in existing algorithms on top of grayscale images. This comes at the cost of including pipeline buffering (latency) due to event collection over some time window, loosing the timestamp information, and potentially needing external IMU integration for long-term odometry.

Many techniques avoid the reconstruction of a full intensity image over a long buffer, but still rely on machine learning methods made for image data, such as Convolutional Neural Networks [12, 18], and thus require formatting events into a sparse 2D grid structure. This has been applied to optical flow estimation [34, 8], object detection [7, 8], and depth estimation [32]. Various aggregation schemes can be used, such as time-window binning or voxel volumes. Different grid sampling schemes are proposed in [13] and [8]. Advantages of these methods include compatibility with image-based learning algorithms, but disadvantages include, once again, inefficiency over sparse grids, loss of precise event timings, and a delay required to collect frames over time windows.

A distinct approach, evaluated on image classification, samples events until they form a connected graph, with a combination of spatial and temporal distances as a measure of edge length [33]. A neural network able to work on graph data [5] is then used to process the inputs. The use of spatial graph convolutions addresses the issue of sparsity found in grid-based approaches but still requires to collect data over a time window.

Per-event methods. Since event-cameras are considered a neuromorphic system, researchers theorized they would go hand-in-hand with a more biologically-grounded model for processing. Spiking Neural Networks (SNNs) [21] are a class of neural networks based on human-vision perception principles, asynchronously activating specific neurons. This makes them a theoretical candidate for processing DVS events, one at a time [2, 24]. In their original form, SNNs are non-differentiable and thus incompatible with backpropagation-based training; therefore, most SNN methods require either proxy-based procedures [31] or an approximation of the original SNN formulation [19]. Nevertheless, these models tend to have lower performance than more modern methods.

Another clear choice for event-by-event classification are RNNs [11], neural networks specifically designed to handle sequential data. Such models, however, usually assume evenly-spaced series inputs, therefore neglecting one of the main features of DVS sensors. To address this, an extension of the LSTM [15] architecture, named PhasedLSTM [22], was devised. This model added time gates to the previous and current intermediate hidden states. These gates open cyclically, modulated by the current input timestamp. PhasedLSTM was tested on event classification, using an embedding for the event coordinates, showing an improvement over LSTM for performance on the same task. Recent approaches process events with recursive strategies [29].

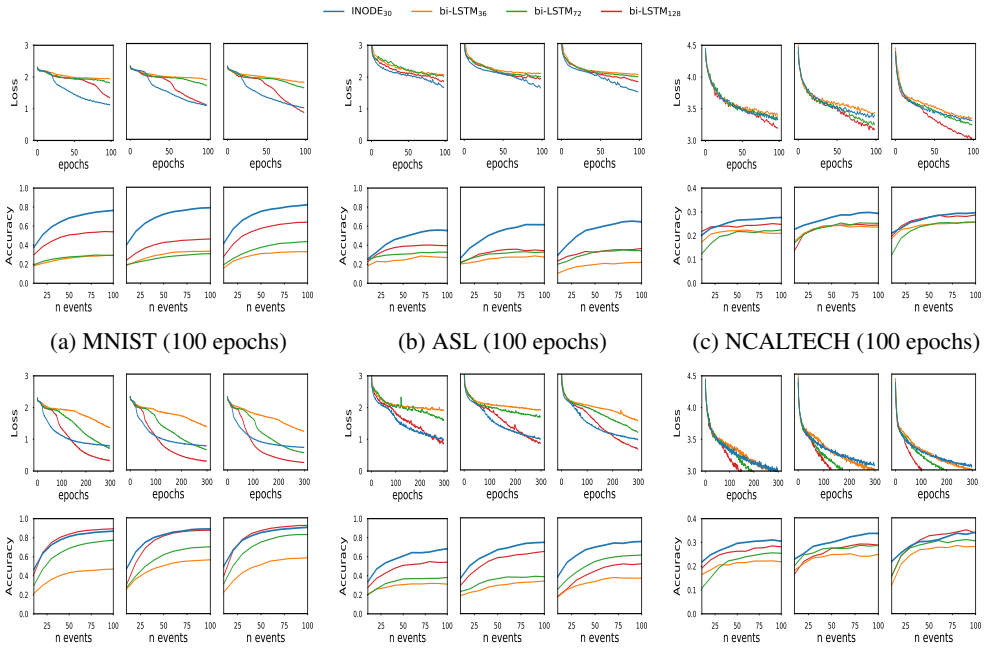
Neural ODEs. NODEs are a recent methodology for modeling data as a dynamical system, governed by a neural network and solved using traditional ODE solvers [9]. Inference is performed using gradient-based optimization through several time steps of the discretized ODE, typically using *explicit* time-stepping schemes [6]. To reduce memory requirements, researchers have proposed using the adjoint method [9, 14]. NODEs have been applied to the time-series domain [28], by employing an LSTM to preprocess irregularly-spaced samples before feeding it into a NODE solver. This adds flexibility to the original formulation, at the cost of additional parameters and increased processing time. Moreover, there is high risk that the conditioning network could perform most of the inference and therefore the NODE results only in an integration task. In this work, we instead consider ODEs with an input connection, similarly to the SNODE architecture in [25].

B Algorithm

Algorithm 1 INODE

Inputs: $\mathbf{e}$, $\mathrm{d}_{\max}$, d_q, $S \ll M$
repeat
  Sample $\{u_i, t_i\}_{i=s}^{s+S}$ from $\mathbf{e}$
  for $i = 0$ **to** $S - 1$ **do**
    $\mathrm{d}\tau = \min((t_{i+1} - t_i)/\mathrm{d}_q, \mathrm{d}_{\max})$
    $h(t_{i+i}) = h(t_i) + \mathrm{d}\tau\ f(h(t_i), u(t_i))$
    $z(t_{i+1}) = g(h(t_{i+1}))$
    $L_{i+1} = L(z(t_{i+1}), \bar{z})$
  end for
  $\mathcal{L} = \frac{1}{S}\sum_{i=1}^{S} L_i$
  $\theta \leftarrow \nabla_\theta \mathcal{L}$
until Convergence

C Tables and Figures

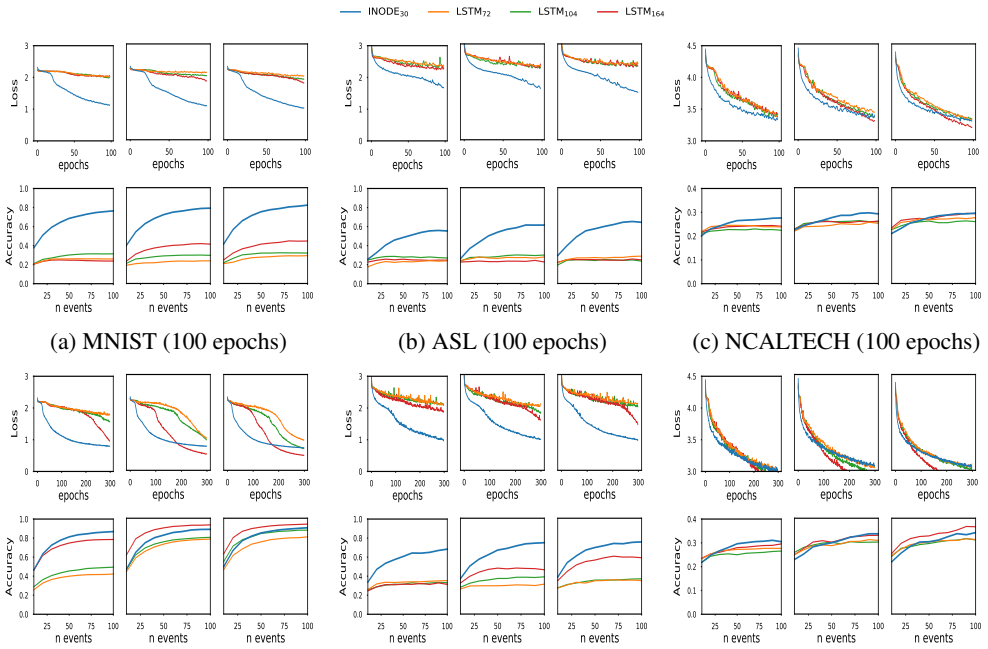


(a) MNIST (100 epochs) (b) ASL (100 epochs) (c) NCALTECH (100 epochs)

(d) MNIST (300 epochs) (e) ASL (300 epochs) (f) NCALTECH (300 epochs)

Figure 4: Summary of results. Train/test losses and classification performance for INODE and multiple bi-LSTM baselines, with increasing number of inference events per digit from 10 to 100. The three images for each dataset sub-figure correspond to training-set fraction of 20% (left), 40% (center), and 100% (right).

The number of states, parameters, and input features for each model are summarized in Table 2.



(a) MNIST (100 epochs) (b) ASL (100 epochs) (c) NCALTECH (100 epochs)

(d) MNIST (300 epochs) (e) ASL (300 epochs) (f) NCALTECH (300 epochs)

Figure 5: Summary of results. Train/test losses and classification performance for INODE and multiple LSTM baselines, with increasing number of inference events per digit from 10 to 100. The three images for each dataset sub-figure correspond to training-set fraction of 20% (left), 40% (center), and 100% (right).

Table 2: Models setup and complexity

MODEL	N STATES	N PARAMS	INPUT
INODE$_{30}$ (OURS)	30	42,161	(x, y, p)
LSTM$_{164}$	164	111,520	(x, y, p, t)
P-LSTM$_{164}$	164	111,192	(x, y, p)
LSTM$_{104}$	104	45,760	(x, y, p, t)
P-LSTM$_{104}$	104	45,552	(x, y, p)
LSTM$_{72}$	72	22,464	(x, y, p, t)
P-LSTM$_{72}$	72	22,320	(x, y, p)
BI-LSTM$_{128}$	128	137,216	(x, y, p, t)
P-BI-LSTM$_{128}$	128	136,704	(x, y, p)
BI-LSTM$_{72}$	72	44,928	(x, y, p, t)
P-BI-LSTM$_{72}$	72	44,640	(x, y, p)
BI-LSTM$_{36}$	36	12,096	(x, y, p, t)
P-BI-LSTM$_{36}$	36	11,952	(x, y, p)

Table 3: f parameterization for INODE and classifier.

	FC$_1$	FC$_u$	FC$_2$	FC$_3$	FC$_c$
INPUT DIM	30	3(+1)	256	128	30
OUTPUT DIM	128	128	128	30	N CLASSES

Table 4: Classification accuracy on NMNIST test set increasing the number of events (10 classes).

MODEL	DATASET %	N EVENTS TEST				
		10	20	30	40	100
INODE$_{30}$	100	0.48	0.66	0.75	0.80	0.89
LSTM$_{164}$	100	**0.63**	**0.80**	**0.86**	**0.89**	**0.94**
LSTM$_{104}$	100	0.55	0.71	0.78	0.81	0.88
P-LSTM$_{104}$	100	0.27	0.29	0.27	0.24	0.18
LSTM$_{72}$	100	0.46	0.61	0.68	0.73	0.81
BI-LSTM$_{128}$	100	0.39	0.66	0.77	0.84	0.93
P-BI-LSTM$_{128}$	100	0.28	0.34	0.39	0.44	0.55
BI-LSTM$_{72}$	100	0.31	0.50	0.62	0.70	0.84
P-BI-LSTM$_{72}$	100	0.26	0.32	0.36	0.40	0.51
BI-LSTM$_{36}$	100	0.22	0.34	0.43	0.48	0.61
P-BI-LSTM$_{36}$	100	0.24	0.30	0.32	0.35	0.44
INODE$_{30}$	40	0.46	0.65	0.73	0.79	0.88
LSTM$_{164}$	40	**0.61**	**0.79**	**0.85**	**0.88**	**0.93**
LSTM$_{104}$	40	0.46	0.62	0.69	0.73	0.80
P-LSTM$_{104}$	40	0.24	0.26	0.24	0.21	0.15
LSTM$_{72}$	40	0.44	0.59	0.65	0.70	0.78
BI-LSTM$_{128}$	40	0.30	0.53	0.68	0.77	0.89
P-BI-LSTM$_{128}$	40	0.27	0.33	0.37	0.41	0.51
BI-LSTM$_{72}$	40	0.27	0.39	0.49	0.56	0.72
P-BI-LSTM$_{72}$	40	0.25	0.30	0.34	0.37	0.45
BI-LSTM$_{36}$	40	0.25	0.36	0.42	0.47	0.58
P-BI-LSTM$_{36}$	40	0.23	0.27	0.30	0.32	0.40
INODE$_{30}$	20	**0.46**	0.63	0.73	0.78	0.87
LSTM$_{164}$	20	**0.46**	0.62	0.68	0.72	0.79
LSTM$_{104}$	20	0.29	0.36	0.41	0.43	0.49
P-LSTM$_{104}$	20	0.22	0.25	0.23	0.20	0.17
LSTM$_{72}$	20	0.26	0.33	0.36	0.39	0.42
BI-LSTM$_{128}$	20	0.42	**0.64**	**0.75**	**0.80**	**0.90**
P-BI-LSTM$_{128}$	20	0.25	0.30	0.34	0.37	0.47
BI-LSTM$_{72}$	20	0.30	0.47	0.58	0.64	0.77
P-BI-LSTM$_{72}$	20	0.23	0.28	0.30	0.33	0.41
BI-LSTM$_{36}$	20	0.21	0.30	0.36	0.40	0.49
P-BI-LSTM$_{36}$	20	0.21	0.24	0.26	0.28	0.34
RANDOM		0.10	0.10	0.10	0.10	0.10

Table 5: Classification accuracy on ASL test set increasing the number of events (24 classes).

MODEL	DATASET %	N EVENTS TEST				
		10	20	30	40	100
INODE$_{30}$	100	**0.37**	**0.51**	**0.61**	**0.67**	**0.79**
LSTM$_{164}$	100	0.35	0.44	0.51	0.55	0.59
P-LSTM$_{164}$	100	0.22	0.25	0.25	0.24	0.21
LSTM$_{104}$	100	0.27	0.31	0.32	0.34	0.37
P-LSTM$_{104}$	100	0.21	0.21	0.23	0.22	0.20
LSTM$_{72}$	100	0.27	0.30	0.33	0.32	0.35
P-LSTM$_{72}$	100	0.24	0.26	0.27	0.28	0.24
BI-LSTM$_{128}$	100	0.18	0.26	0.35	0.40	0.54
P-BI-LSTM$_{128}$	100	0.28	0.33	0.36	0.41	0.47
BI-LSTM$_{72}$	100	0.25	0.36	0.43	0.49	0.61
P-BI-LSTM$_{72}$	100	0.29	0.32	0.36	0.38	0.45
BI-LSTM$_{36}$	100	0.17	0.25	0.29	0.32	0.38
P-BI-LSTM$_{36}$	100	0.23	0.27	0.30	0.31	0.36
INODE$_{30}$	40	**0.36**	**0.50**	**0.58**	**0.64**	**0.69**
LSTM$_{164}$	40	0.32	0.39	0.44	0.47	0.46
P-LSTM$_{164}$	40	0.19	0.19	0.19	0.19	0.18
LSTM$_{104}$	40	0.28	0.32	0.34	0.36	0.39
P-LSTM$_{104}$	40	0.18	0.19	0.20	0.19	0.21
LSTM$_{72}$	40	0.26	0.29	0.29	0.30	0.31
P-LSTM$_{72}$	40	0.24	0.26	0.25	0.27	0.25
BI-LSTM$_{128}$	40	0.29	0.40	0.48	0.55	0.65
P-BI-LSTM$_{128}$	40	0.27	0.32	0.35	0.37	0.41
BI-LSTM$_{72}$	40	0.23	0.26	0.31	0.35	0.40
P-BI-LSTM$_{72}$	40	0.23	0.28	0.30	0.33	0.36
BI-LSTM$_{36}$	40	0.19	0.22	0.24	0.28	0.34
P-BI-LSTM$_{36}$	40	0.23	0.27	0.30	0.31	0.35
INODE$_{30}$	20	**0.32**	**0.47**	**0.55**	**0.60**	**0.71**
LSTM$_{164}$	20	0.25	0.29	0.31	0.31	0.31
P-LSTM$_{164}$	20	0.17	0.17	0.16	0.16	0.15
LSTM$_{104}$	20	0.26	0.29	0.31	0.32	0.33
P-LSTM$_{104}$	20	0.19	0.18	0.19	0.19	0.17
LSTM$_{72}$	20	0.26	0.32	0.34	0.33	0.35
P-LSTM$_{72}$	20	0.19	0.19	0.16	0.17	0.18
BI-LSTM$_{128}$	20	0.28	0.37	0.43	0.48	0.55
P-BI-LSTM$_{128}$	20	0.25	0.28	0.30	0.34	0.37
BI-LSTM$_{72}$	20	0.20	0.26	0.32	0.34	0.39
P-BI-LSTM$_{72}$	20	0.24	0.28	0.30	0.30	0.35
BI-LSTM$_{36}$	20	0.21	0.26	0.28	0.30	0.33
P-BI-LSTM$_{36}$	20	0.23	0.26	0.27	0.28	0.31
RANDOM		0.04	0.04	0.04	0.04	0.04

Table 6: Classification accuracy on NCALTECH test set increasing the number of events (100 classes).

MODEL	DATASET %	N EVENTS TEST				
		10	20	30	40	100
INODE$_{30}$	100	0.22	0.26	0.29	0.30	0.34
LSTM$_{164}$	100	**0.25**	**0.29**	**0.32**	**0.32**	**0.36**
P-LSTM$_{164}$	100	0.22	0.24	0.24	0.24	0.21
LSTM$_{104}$	100	0.24	0.27	0.28	0.29	0.31
P-LSTM$_{104}$	100	0.23	0.25	0.25	0.24	0.21
LSTM$_{72}$	100	0.24	0.27	0.29	0.30	0.31
P-LSTM$_{72}$	100	0.23	0.24	0.23	0.24	0.24
BI-LSTM$_{128}$	100	0.16	0.24	0.28	0.31	0.35
P-BI-LSTM$_{128}$	100	0.21	0.24	0.26	0.26	0.28
BI-LSTM$_{72}$	100	0.16	0.24	0.28	0.29	0.30
P-BI-LSTM$_{72}$	100	0.21	0.24	0.25	0.26	0.28
BI-LSTM$_{36}$	100	0.12	0.21	0.24	0.27	0.28
P-BI-LSTM$_{36}$	100	0.21	0.23	0.24	0.25	0.26
INODE$_{30}$	40	0.23	0.27	0.29	**0.31**	**0.34**
LSTM$_{164}$	40	0.25	0.27	**0.30**	**0.31**	0.33
P-LSTM$_{164}$	40	0.22	0.23	0.22	0.22	0.20
LSTM$_{104}$	40	**0.26**	**0.28**	0.29	0.30	0.30
P-LSTM$_{104}$	40	0.21	0.22	0.22	0.22	0.22
LSTM$_{72}$	40	0.25	0.27	0.28	0.29	0.31
P-LSTM$_{72}$	40	0.20	0.21	0.20	0.21	0.20
BI-LSTM$_{128}$	40	0.17	0.22	0.24	0.25	0.29
P-BI-LSTM$_{128}$	40	0.22	0.24	0.26	0.26	0.28
BI-LSTM$_{72}$	40	0.20	0.24	0.25	0.27	0.28
P-BI-LSTM$_{72}$	40	0.21	0.23	0.24	0.25	0.27
BI-LSTM$_{36}$	40	0.18	0.21	0.23	0.24	0.25
P-BI-LSTM$_{36}$	40	0.20	0.21	0.22	0.23	0.25
INODE$_{30}$	20	0.22	**0.25**	0.26	**0.28**	**0.30**
LSTM$_{164}$	20	**0.24**	**0.25**	0.26	0.27	**0.30**
P-LSTM$_{164}$	20	0.21	0.22	0.22	0.22	0.20
LSTM$_{104}$	20	0.22	0.24	0.25	0.25	0.27
P-LSTM$_{104}$	20	0.20	0.23	0.22	0.23	0.21
LSTM$_{72}$	20	0.23	0.25	**0.27**	0.27	0.28
P-LSTM$_{72}$	20	0.19	0.20	0.20	0.20	0.20
BI-LSTM$_{128}$	20	0.19	0.23	0.25	0.26	0.28
P-BI-LSTM$_{128}$	20	0.21	0.23	0.25	0.26	0.26
BI-LSTM$_{72}$	20	0.11	0.15	0.20	0.22	0.25
P-BI-LSTM$_{72}$	20	0.20	0.21	0.23	0.24	0.25
BI-LSTM$_{36}$	20	0.17	0.18	0.21	0.20	0.22
P-BI-LSTM$_{36}$	20	0.20	0.20	0.23	0.23	0.24
RANDOM		0.01	0.01	0.01	0.01	0.01