A Differentiable Recurrent Surface for Asynchronous Event-Based Data *Supplementary Material*

Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci

Politecnico di Milano, Italy
{marco.cannici,marco.ciccone,andrea.romanoni,matteo.matteucci}@polimi.it

1 Implementation

The Matrix-LSTM feature extraction process can be implemented efficiently by means of two order-aware reshape operations. These two operations, namely *groupByPixel* and *groupByTime*, allow event streams to be split based on the pixel location and temporal bin. After being reshaped, the input is ready to be processed by a single LSTM network, implementing parameter sharing across all pixel locations and temporal windows. In the following we give a detailed overview of the two reshape operators.

1.1 GroupByPixel

This operation translates from event-sequences to pixel-sequences. Let X be a tensor of shape $N \times T_{max} \times F$, representing the features $f_{n,i}^{(x,y)}$ of a batch of N samples, where T_{max} is the length of the longest sequence in the batch. We define the *groupByPixel* mapping on X as an order-aware reshape operation that rearranges the events into a tensor of pixel-sequences of shape $P \times T_{max}^{(x,y)} \times F$ where $T_{max}^{(x,y)}$ is the length of the longest pixel sequence $\mathcal{E}_n^{(x,y)}$ and P is the number of active pixels (i.e., having at least one event) in the batch, which equals $N \cdot H \cdot W$ only in the worst case. Pixel-sequences shorter than $T_{max}^{(x,y)}$ are padded with zero events to be processed in parallel.

The tensor thus obtained is then processed by the LSTM cell that treats samples in the first dimension independently, effectively implementing parameter sharing and applying the transformation in parallel over all the pixels. The LSTM output tensor, which has the same shape of the input one, is then sampled by taking the output corresponding to the last event in each pixel-sequence $\mathcal{E}_n^{(x,y)}$, ignoring values computed on padded values, and the obtained values are then used to populate the dense representation. To improve efficiency, for each pixel-sequence $\mathcal{E}_n^{(x,y)}$, *groupByPixel* keeps also track of the original spatial position (x, y), the index of the sample inside the batch and the length of the pixel-sequence $T_n^{(x,y)}$, namely the index of the last event before padding. Given this set of indexes, the densification step can be performed as a simple slicing operation. See Figure 1 for visual clues. *groupByPixel* is implemented as a custom CUDA

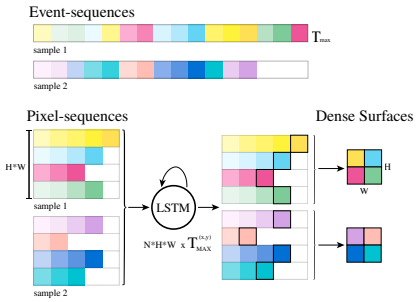

Fig. 1. An example of the *groupByPixel* operation on a batch of $N = 2$ samples and a 2×2 pixel resolution. Different colors refer to different pixel locations while intensity indicates time. For clarity, the features dimension is not shown in the figure

Table 1. Comparison between Matrix-LSTM and ConvLSTM on both Ev2Vid and EST ResNet18 configurations on the N-Cars dataset

		delay relative		ts absolute	
		3×3	5×5	3×3	5×5
Ev2Vid with 3 chans, 1 bin	Matrix-LSTM (ours)	$\mathbf{95.05 \pm 0.96\%}$	$\mathbf{93.38 \pm 0.64\%}$	$\mathbf{94.92 \pm 0.74\%}$	$\mathbf{94.34 \pm 0.94\%}$
	ConvLSTM [7]	$92.33 \pm 0.41\%$	$92.65 \pm 0.78\%$	$93.97 \pm 1.30\%$	$93.61 \pm 1.59\%$
EST with 16 chans, 1 bin	Matrix-LSTM (ours)	$\mathbf{93.14 \pm 0.77\%}$	$\mathbf{92.18 \pm 0.28\%}$	$\mathbf{92.83 \pm 1.32\%}$	$\mathbf{92.15 \pm 0.67\%}$
	ConvLSTM [7]	$90.39 \pm 0.94\%$	$90.73 \pm 1.05\%$	$92.52 \pm 1.26\%$	$92.05 \pm 0.56\%$

kernel that processes each sample in parallel and places each event feature in the output tensor maintaining the original temporal order.

1.2 GroupByTime

The Matrix-LSTM variant that operates on temporal bins performs a similar pre-processing step. Each sample in the batch is divided into a fixed set of intervals. The *groupByTime* cuda kernel pre-processes the input events generating a $N * B \times T^b_{max} \times F$ tensor where the B bins are grouped in the first dimension and taking care of properly padding intervals (T^b_{max} is the length of the longest bin in the batch). The Matrix-LSTM mechanism is then applied as usual and the resulting $N * B \times H \times W \times C$ tensor is finally reshaped into a $N \times H \times W \times B * C$ event-surface.

2 Matrix-LSTM vs ConvLSTM

In Figure 2a of the paper we report a comparison between ConvLSTM and Matrix-LSTM using the Ev2Vid–ResNet18 configuration, i.e., the configuration of choice for all comparisons in the paper. For completeness, in Table 1 we extend the evaluation also to EST, using the 16 channels and 1 bin setting, which is one of our best performing EST configuration on N-Cars. Matrix-LSTM performs better on all experiments, highlighting its capabilities to better handle asynchronous event-based data if compared to ConvLSTM. We also highlight

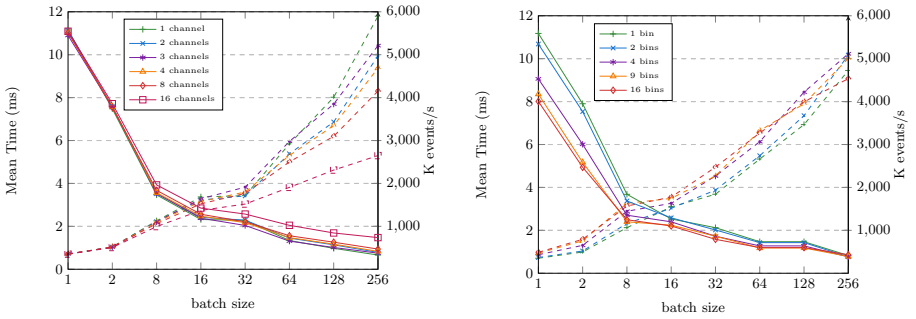

Fig. 2. Number of processed events per second (dashed lines) and timing (solid lines) with varying number of channels **(a)**, and bins **(b)**

that the performance improvement is greater on *delay relative* temporal features than on *ts absolute* ones. ConvLSTM, indeed, processes temporal slices containing potentially uncorrelated events that happened at different time instants. While delays are always consistent within each pixel sequence, they are not within the ConvLSTM kernel receptive field. Using an absolute temporal encoding alleviates this issue on both Ev2Vid and EST architectures, while still performing worst than Matrix-LSTM. Our extraction layer, indeed, preserves the original events arrival order within each receptive field during features extraction, which allows to achieve better performance both on *ts absolute* and *delay relative* input features. Moreover, structured *delay relative* features perform better on Matrix-LSTM than simple absolute features.

3 Time Performance

While the performance reported in Figure 3b of the paper are computed on each sample independently to enable a fair comparison with the other methods, in Figure 2a and Figure 2b we study instead how the mean time required to process a sample over all the N-Cars training dataset and the corresponding events throughput change as a function of the batch size. Both performance dramatically increase when multiple samples are processed simultaneously in batch. This is crucial at training time, when optimization techniques greatly benefit from batch computation.

Furthermore, while increasing the number of output channels, for the same choice of batch size, increases the time required to process each sample (since the resulting Matrix-LSTM operates on a larger hidden state), increasing the number of bins has an opposite behaviour. Multiple intervals are indeed processed independently and in parallel by the Matrix-LSTM that has to process a smaller number of events in each spatial location, sequentially. In both configurations, finally, increasing the batch size reduces the mean processing time.

4 Classification

Table 2. Classification accuracy (%) on the N-MNIST [5] dataset.

Method	Classifier	Channels (bins)	Accuracy
H-First	spike-based	-	
HOTS [4]	histogram similarity	-	80.8
HATS [8]	SVM	-	99.1
G-CNN [1]	Graph CNN	-	98.5
RG-CNN [1]	Graph CNN	-	**99.0**
Events Count [1]	ResNet50	2 (1)	98.4
Ev2Vid [6]	Ev2Vid custom convnet	1 (1)	98.3
Matrix-LSTM (Ours)	Ev2Vid custom convnet	1 (1)	$\mathbf{98.9 \pm 0.21}$

Table 3. Classification accuracy (%) on the ASL-DVS [1] dataset.

Method	Classifier	Channels (bins)	Accuracy
G-CNN [1]	Graph CNN	-	87.5
RG-CNN [1]	Graph CNN	-	90.1
Events Count [1]	ResNet50	2 (1)	88.6
EST [2]	ResNet50	2 (1)	99.57
Matrix-LSTM (Ours)	ResNet50	2 (1)	$\mathbf{99.73 \pm 0.04}$

We perform additional experiments on the N-MNIST dataset [5] and on the newly introduced ASL-DVS [1] dataset. On N-MNIST we directly compare with the Ev2Vid [6] reconstruction procedure, where the custom convolutional network proposed in [6] is used as backbone, while we compare with the EST [2] surface on ASL-DVS, making use of ResNet50 [3] as backbone. On both cases, Matrix-LSTM performs better than other event-surface mechanisms and also outperforms alternative classification architectures.

5 Optical Flow Prediction

5.1 Ev-FlowNet Baseline Results

We performed optical flow experiments starting from the publicly available Ev-FlowNet codebase [9] and replacing the original hand-crafted features with the proposed Matrix-LSTM layer. We first made sure to revert the baseline architecture to the original configuration, checking that we were able to replicate the paper results. Indeed, the public code contains minor upgrades over the paper

Table 4. Effect of adding a Squeeze-and-Excitation layer on the optical flow prediction task

Method		*indoor_flying1*				*indoor_flying2*				*indoor_flying3*			
		dt=1		*dt=4*		*dt=1*		*dt=4*		*dt=1*		*dt=4*	
		AEE	%Outlier	AEE	%Outlier	AEE	%Outlier	AEE	%Outlier	AEE	%Outlier	AEE	%Outlier
Ev-FlowNet [10]	-	1.03	2.2	2.25	24.7	2.12	15.1	4.05	45.3	1.53	11.9	3.45	39.7
Ev-FlowNet (ours)	-	1.015	2.736	3.432	48.685	1.606	12.089	5.957	63.226	1.548	11.937	5.247	57.662
Matrix-LSTM (Ours)	1 bin	1.017	2.071	3.366	42.022	1.642	13.89	5.870	65.379	1.432	10.44	5.015	57.094
	2 bins	0.829	**0.471**	**2.269**	**23.558**	1.194	**5.341**	**3.946**	**42.450**	1.083	**4.390**	**3.172**	**31.975**
	2 bins + SELayer	**0.821**	0.534	2.378	25.995	**1.191**	5.590	4.333	45.396	**1.077**	4.805	3.549	36.822
	4 bins	0.969	1.781	3.023	36.085	1.505	11.63	4.870	49.077	1.507	12.97	4.652	43.267
	4 bins + SELayer	**0.844**	**0.634**	**2.330**	**24.777**	**1.213**	**6.057**	**4.322**	**44.769**	**1.070**	**4.625**	**3.588**	**36.442**
	8 bins	**0.881**	**0.672**	**2.290**	**24.203**	1.292	**6.594**	**3.978**	**42.230**	1.181	5.389	**3.346**	**33.951**
	8 bins + SELayer	0.905	0.885	2.308	24.597	**1.286**	6.761	4.046	44.366	**1.177**	**5.318**	3.391	35.452

version. We contacted the authors that provided us with the needed modifications. These consist of removing the batch normalization layers, setting to 2 the number of output channels of the layer preceding the optical flow prediction layer, and disabling random rotations during training. For completeness, we report the results we obtained by training the baseline from scratch with these fixes in Table 4.

To test how the network adapts to different flow magnitudes, the Ev-FlowNet [10] was tested on two evaluation settings for each test sequence: with input frames and corresponding events that are one frame apart (denoted as *dt=1*), and with frames and events four frames apart (denoted as *dt=4*). While we were able to closely replicate the results of the first configuration (*dt=1*), with a minor improvement in the *indoor_flying2* sequence, the performance we obtain on the *dt=4* setup is instead worse on all sequences, as reported on the first two rows of Table 4.

Despite this discrepancy, which prevents the Matrix-LSTM performance on *dt=4* settings to be directly compared with the results reported on the Ev-FlowNet paper, we can still evaluate the benefits of our surface on larger flow magnitudes. Indeed, this work evaluates the Matrix-LSTM layer based on the relative performance improvement obtained by substituting the original features with our layer. Using our Ev-FlowNet results as baseline, we show that Matrix-LSTM is able to improve the optical flow quality even on the *dt=4* setting, highlighting the capability of the layer to adapt to different sequence lengths and movement conditions. We report an improvement of up to 30.426% on *dt=1* settings and up to 39.546% on *dt=4* settings using our results as baseline.

5.2 Squeeze-and-Excitation Layer

Optical flow prediction is a complex task that requires neural networks to extract accurate features precisely describing motion inside the scene. An event aggregation mechanism is therefore required to extract rich temporal features from the events. In Section 4.2 of the paper we show that time resolution is a key factor for extracting effective feature with Matrix-LSTM. In particular, increasing the number of bins has great impact on the predicted flow and allows the network to retain temporal information over long sequences. Here we focus,

instead, on the effect of correlating temporal features by adding a SELayer to the Matrix-LSTM output. Table 4 reports the performance obtained using this additional layer on the MVSEC [11] task. The results we obtained show that adding an SELayer only improves performance on the 4 bins configuration for the *dt=4* benchmark, while it consistently helps reducing the *AEE* metric on the *dt=1* setting.

By comparing features obtained from subsequent intervals, the SELayer adaptively recalibrates features and helps modelling interdependencies between time instants, which is crucial for predicting optical flow. We believe that a similar approach can also be applied to other event aggregation mechanisms based on voxel-grids of temporal bins to improve their performance, especially those employing data driven optimization mechanisms [2].

6 Qualitative Results

The event aggregation process performed by the Matrix-LSTM layer is incremental. Events in each pixel location are processed sequentially; state and output of the LSTM are updated each time. We propose to visualize the Matrix-LSTM surface as an RGB image by using the ResNet18-Ev2Vid configuration and interpreting the 3 output channels as RGB color. A video of such visualization showing the incremental frame reconstruction on N-Caltech101 samples is provided at this url: https://marcocannici.github.io/matrixlstm.

We use a similar visualization technique to show optical flow predictions for *indoor_flying* sequences. Since we use our best performing model that uses 2 temporal bins, we decide to only show the first 3 channels of each temporal interval. Moreover, instead of visualizing how the event representation builds as new events arrive, we only show the frame obtained after having processed each window of events.

References

1. Bi, Y., Chadha, A., Abbas, A., Bourtsoulatze, E., Andreopoulos, Y.: Graph-based object classification for neuromorphic vision sensing. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 491–501 (2019)
2. Gehrig, D., Loquercio, A., Derpanis, K.G., Scaramuzza, D.: End-to-end learning of representations for asynchronous event-based data. In: IEEE International Conference of Computer Vision (ICCV) (October 2019)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
4. Lagorce, X., Orchard, G., Galluppi, F., Shi, B.E., Benosman, R.B.: Hots: a hierarchy of event-based time-surfaces for pattern recognition. IEEE transactions on pattern analysis and machine intelligence **39**(7), 1346–1359 (2016)
5. Orchard, G., Jayawant, A., Cohen, G.K., Thakor, N.: Converting static image datasets to spiking neuromorphic datasets using saccades. Frontiers in neuroscience **9**, 437 (2015)
6. Rebecq, H., Ranftl, R., Koltun, V., Scaramuzza, D.: Events-to-video: Bringing modern computer vision to event cameras. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3857–3866 (2019)
7. SHI, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.k., WOO, W.c.: Convolutional lstm network: A machine learning approach for precipitation nowcasting. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28, pp. 802–810. Curran Associates, Inc. (2015)
8. Sironi, A., Brambilla, M., Bourdis, N., Lagorce, X., Benosman, R.: Hats: Histograms of averaged time surfaces for robust event-based object classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1731–1740 (2018)
9. Zhu, A., Yuan, L., Chaney, K., Daniilidis, K.: Ev-flownet: Self-supervised optical flow estimation for event-based cameras. https://github.com/daniilidis-group/EV-FlowNet
10. Zhu, A., Yuan, L., Chaney, K., Daniilidis, K.: Ev-flownet: Self-supervised optical flow estimation for event-based cameras. In: Proceedings of Robotics: Science and Systems. Pittsburgh, Pennsylvania (June 2018)
11. Zhu, A.Z., Thakur, D., Özaslan, T., Pfrommer, B., Kumar, V., Daniilidis, K.: The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. IEEE Robotics and Automation Letters **3**(3), 2032–2039 (2018)