Supplementary Material: Event-based Asynchronous Sparse Convolutional Networks

Nico Messikommer*, Daniel Gehrig*, Antonio Loquercio, and Davide Scaramuzza

Dept. Informatics, Univ. of Zurich and
Dept. of Neuroinformatics, Univ. of Zurich and ETH Zurich

1 Supplementary Material

In the supplementary material, references which point to the main manuscript will be referenced with a leading "M-". In Sec. 1.1 we describe an efficient recursive method for computing the rulebook $\mathcal{R}_{\mathbf{k},n}$ in Eq. (M-7) and present the asynchronous propagation of changes through events in algorithmic form in Tab. 1. In Sec. 1.2 we present a proof of the equivalence of network outputs using asynchronous and synchronous networks. In Sec. 1.3 we present additional details about the input representations and FLOP calculations used in the experiments in Sec. M-4. In Sec. 1.4 we present a sensitivity analysis where we vary the number of events used for training and justify our choice of 25'000 events for all experiments. Finally, in Sec. 1.5 we show additional qualitative object detection results.

1.1 Efficient Rulebook Update

At each layer the rulebook $\mathcal{R}_{\mathbf{k},n}$ and receptive field $\mathcal{F}_n$ are

$$\begin{aligned}\mathcal{F}_n &= \{\mathbf{i}-\mathbf{k} | \mathbf{i} \in \mathcal{F}_{n-1} \text{ and } \mathbf{k} \in \mathcal{K}_{n-1} \text{ if } \mathbf{i}-\mathbf{k} \in \mathcal{A}_t\} \\ \mathcal{R}_{\mathbf{k},n} &= \{(\mathbf{i}, \mathbf{i}-\mathbf{k}) | \mathbf{i} \in \mathcal{F}_{n-1} \text{ if } \mathbf{i}-\mathbf{k} \in \mathcal{A}_t\}.\end{aligned}$$

At the input these are initialized as $R_{\mathbf{k},0} = \emptyset$ and $\mathcal{F}_0 = \{\mathbf{u}_i'\}$. The propagation of these two data structures is illustrated in Fig. M-2. We observe that at each layer the rulebook and receptive field can be computed by reusing the data from the previous layer. We can do this by decomposing the receptive field into a frontier set f_n (Fig. M-2 (a) magenta sites)and visited state set F_n (Fig. M-2 (a) green sites). At each layer $\mathcal{F}_n = f_n \cup F_n$. To efficiently update both $\mathcal{F}_n$ and $\mathcal{R}_{\mathbf{k},n}$ at each layer we only consider the rules that are added due to inputs in the frontier set (Fig. M-2 (a), magenta lines). These can be appended to the existing rulebook. In addition, the receptive field $\mathcal{F}_n$ can be updated similarly, by adding new update sites reached from the frontier set. This greatly reduces the sites that need to be considered in the computation of $\mathcal{R}_{\mathbf{k},n}$ and $\mathcal{F}_n$ in Eqs. (M-6) and (M-7).

* Equal contribution

1.2 Equivalence of Synchronous and Asynchronous Updates

We start with Eq. (M-4), which we repeat here:

$$\tilde{y}^t_{n+1}(\mathbf{u}, c) = \begin{cases} b_n(c) + \sum_{c'} \sum_{\mathbf{k} \in \mathcal{K}_n} \sum_{(\mathbf{i},\mathbf{u}) \in R_{t,\mathbf{k}}} W_n(\mathbf{k}, c', c) y^t_n(\mathbf{i}, c'), & \text{for } \mathbf{u} \in \mathcal{A}_t \\ 0 & \text{else} \end{cases}$$
$$y^t_{n+1} = \sigma(\tilde{y}^t_{n+1})$$

Here the input layer is $y^t_0(\mathbf{u}, c) = H_{t_N}(\mathbf{u}, c)$. In a next step we assume changes to the input layer as in Eq. (M-3). These changes occur at sites $\mathbf{u}_i \in \mathcal{F}_0$ with magnitude $\Delta_i(c) = y^{t+1}_0(\mathbf{u}_i, c) - y^t_0(\mathbf{u}_i, c)$. The sites $\mathbf{u}_i$ can be categorized into three groups: sites that are and remain active, sites that become inactive (feature becomes 0) and sites that become active. We will now consider how $y^{t+1}_n(\mathbf{u}, c)$ evolves:

$$\tilde{y}^{t+1}_1(\mathbf{u}, c) = b_0(c) + \sum_{c'} \sum_{\mathbf{k} \in \mathcal{K}_0} \sum_{(\mathbf{i},\mathbf{u}) \in R_{t+1,\mathbf{k}}} W_n(\mathbf{k}, c', c) y^{t+1}_0(\mathbf{i}, c') \tag{1}$$
$$= b_0(c) + \sum_{c'} \sum_{\mathbf{k} \in \mathcal{K}_0} \sum_{(\mathbf{i},\mathbf{u}) \in R_{t+1,\mathbf{k}}} W_0(\mathbf{k}, c', c) \left(y^t_0(\mathbf{i}, c') + \Delta(\mathbf{i}, c') \right) \tag{2}$$
$$\tag{3}$$

Here we define the increment $\Delta_0(\mathbf{u}, c)$. This increment is only non-zero for sites at which the input $y^t_0(\mathbf{i}, c)$ changed, so for $(\mathbf{i}, \mathbf{u}) \in R_{t+1,\mathbf{k}}$ such that $\mathbf{i} \in \mathcal{F}_0$. At time $t+1$ the rulebook $R_{t+1,\mathbf{k}}$ is modified for every site $\mathbf{u}_j$ that becomes newly active:

$$R_{t+1,\mathbf{k}} = R_{t,\mathbf{k}} \cup \bigcup_j \{(\mathbf{u}_j + \mathbf{k}, \mathbf{u}_j) | \mathbf{u}_j + \mathbf{k} \in \mathcal{A}_{t+1}\} \cup \{(\mathbf{u}_j, \mathbf{u}_j - \mathbf{k}) | \mathbf{u}_j - \mathbf{k} \in \mathcal{A}_t\}$$

and every site u_l that becomes inactive

$$R_{t+1,\mathbf{k}} = R_{t,\mathbf{k}} \setminus \bigcup_l \{(\mathbf{u}_l + \mathbf{k}, \mathbf{u}_l) | \mathbf{u}_l + \mathbf{k} \in \mathcal{A}_t\} \cup \{(\mathbf{u}_l, \mathbf{u}_l - \mathbf{k}) | \mathbf{u}_l - \mathbf{k} \in \mathcal{A}_{t+1}\}$$

For both newly active and newly inactive site we may ignore the first term in the union since these correspond to rules that influence the output sites $\mathbf{u}_j$ and $\mathbf{u}_l$. In Fig. M-2 (b) and (c) these rules would correspond to lines leading from input sites (top layer) to the newly active (blue) or newly inactive (white) sites. However, the outputs at these sites can be computed using Eq. (M-4) for newly active sites and simply set to 0 for newly inactive sites, and so we ignore them in updating the next layer. What remains are the contributions of the second term in the union which correspond to the magenta lines in the top layer of Fig. M-2 (b) and (c), which we define as $r_{\mathbf{k},\text{act}}$ and $r_{\mathbf{k},\text{inact}}$ respectively.

If we restrict the output sites $\mathbf{u}$ to be sites that remain active, we may expand Eq. (1) as:

$$\begin{aligned}
\tilde{y}_1^{t+1}(\mathbf{u},c) =& b_0(c) + \sum_{c'} \sum_{\mathbf{k}\in\mathcal{K}_0} \sum_{(\mathbf{i},\mathbf{u})\in R_{t+1,\mathbf{k}}} W_0(\mathbf{k},c',c)\left(y_0^t(\mathbf{i},c') + \Delta(\mathbf{i},c')\right) \\
=& b_0(c) + \sum_{c'} \sum_{\mathbf{k}\in\mathcal{K}_0} \sum_{(\mathbf{i},\mathbf{u})\in R_{t,\mathbf{k}}} W_0(\mathbf{k},c',c)\left(y_0^t(\mathbf{i},c') + \Delta(\mathbf{i},c')\right) \\
&- \sum_{c'} \sum_{\mathbf{k}\in\mathcal{K}_0} \sum_{(\mathbf{i},\mathbf{u})\in r_{\mathbf{k},\text{inact}}} W_0(\mathbf{k},c',c) \underbrace{\left(y_0^t(\mathbf{i},c') + \Delta(\mathbf{i},c')\right)}_{=0 \text{ for } \mathbf{i}=\mathbf{u}_l} \\
&+ \sum_{c'} \sum_{\mathbf{k}\in\mathcal{K}_0} \sum_{(\mathbf{i},\mathbf{u})\in r_{\mathbf{k},\text{act}}} W_0(\mathbf{k},c',c)(\underbrace{y_0^t(\mathbf{i},c')}_{=0 \text{ for } \mathbf{i}=\mathbf{u}_j} + \Delta(\mathbf{i},c'))
\end{aligned}$$

Where we have used the fact that at newly inactive sites $y_0^t(\mathbf{i},c') + \Delta(\mathbf{i},c') = 0$ and at newly active sites $y_0^t(\mathbf{i},c') = 0$. This can be simplified as:

$$\begin{aligned}
\tilde{y}_1^{t+1}(\mathbf{u},c) =& b_0(c) + \sum_{c'} \sum_{\mathbf{k}\in\mathcal{K}_0} \sum_{(\mathbf{i},\mathbf{u})\in R_{t,\mathbf{k}}} W_0(\mathbf{k},c',c)\left(y_0^t(\mathbf{i},c') + \Delta(\mathbf{i},c')\right) \\
&+ \sum_{c'} \sum_{\mathbf{k}\in\mathcal{K}_0} \sum_{(\mathbf{i},\mathbf{u})\in r_{\mathbf{k},\text{act}}} W_0(\mathbf{k},c',c)\Delta(\mathbf{i},c') \\
=& \underbrace{b_0(c) + \sum_{c'} \sum_{\mathbf{k}\in\mathcal{K}_0} \sum_{(\mathbf{i},\mathbf{u})\in R_{t,\mathbf{k}}} W_0(\mathbf{k},c',c) y_0^t(\mathbf{i},c')}_{y_1^t(\mathbf{u},c)} \\
&+ \sum_{c'} \sum_{\mathbf{k}\in\mathcal{K}_0} \sum_{(\mathbf{i},\mathbf{u})\in R_{t,\mathbf{k}}} W_0(\mathbf{k},c',c)\Delta_0(\mathbf{i},c') \\
&+ \sum_{c'} \sum_{\mathbf{k}\in\mathcal{K}_0} \sum_{(\mathbf{i},\mathbf{u})\in r_{\mathbf{k},\text{act}}} W_0(\mathbf{k},c',c)\Delta_0(\mathbf{i},c') \\
=& y_1^t(\mathbf{u},c) + \sum_{c'} \sum_{\mathbf{k}\in\mathcal{K}_0} \sum_{(\mathbf{i},\mathbf{u})\in R_{t,\mathbf{k}}\cup r_{\mathbf{k},\text{act}}} W_0(\mathbf{k},c',c)\Delta_0(\mathbf{i},c')
\end{aligned}$$

It remains to find the rules $(\mathbf{i},\mathbf{u})$ in $R_{t,\mathbf{k}} \cup r_{\mathbf{k},\text{act}}$ which have a non-zero contribution to the sum, *i.e.* for which $\Delta_0(\mathbf{i},c')$ is non-zero. The increment is exactly non-zero for $\mathbf{i} \in \mathcal{F}_0$, corresponding to the input site affected by the new event. Note that this site could either *(i)* remain active (input for rule in $R_{t,\mathbf{k}}$), *(ii)* become inactive (input for rule in $R_{t,\mathbf{k}}$) or *(iii)* become active (input for rule in $r_{\mathbf{k},\text{act}}$)). Therefore, the rules that have a non-zero contribution are the ones drawn as magenta lines in the top row of Fig. M-2 (a), (b) and (c) respectively, where we ignore rules with newly active or inactive sites output sites. These rules also correspond exactly to $\mathcal{R}_{\mathbf{k},1}$ defined in Eq. M-7. The resulting update equation becomes:

$$\tilde{y}_1^{t+1}(\mathbf{u},c) = y_1^t(\mathbf{u},c) + \sum_{c'} \sum_{\mathbf{k}\in\mathcal{K}_0} \sum_{(\mathbf{i},\mathbf{u})\in\mathcal{R}_{\mathbf{k},1}} W_0(\mathbf{k},c',c)\Delta_0(\mathbf{i},c')$$

This is exactly Eq. (M-9). By applying the non-linearity we arrive at Eq. (M-10).

Now let us consider how the update propagates to the next layer. For this we need to find $\mathcal{F}_1$, *i.e.* the input sites of layer 1 that change. These are exactly the updated output sites of layer 0. Every $\mathbf{i} \in \mathcal{F}_0$ affects the output site $\mathbf{u}$ for which $(\mathbf{i}, \mathbf{u}) \in \mathcal{R}_{\mathbf{k},1}$. To be part of this rulebook $\mathbf{u} = \mathbf{i} - \mathbf{k}$ and so we see that $\mathbf{i} - \mathbf{k} \in \mathcal{F}_1$ for all $\mathbf{k} \in \mathcal{K}_0$, which is exactly mirrored by Eq. (M-7).

To propagate updates through layer 1 we thus repeat the steps up until now, but only consider changes at sites $\mathcal{F}_1$ instead of $\mathcal{F}_0$. By iterating this procedure, all layers of the network can be updated. This concludes the proof.

1.3 Representations and FLOP computation

Representations As the proposed asynchronous framework does not require any specific input representation, we evaluate two event embeddings, which are sparse in time and space. The two event representations tested for both tasks are the event histogram [1] and the event queue [2]. The former creates a two-channel image by binning events with positive polarity to the first channel and events with negative polarity to the second channel. This histogram is created for a sliding window containing a constant number of events. Therefore, if an event enters or leaves the sliding window, an update site is created and propagated through the network. The second representation called event queue [2] is applied in a sliding window fashion as well. The event queue stores the timestamps and polarities of the incoming events in a queue at the corresponding image locations. The queues have a fixed length of 15 entries and are initialised with zeros. If a queue is full, the oldest event is discarded. The four dimensional tensor containing the timestamps and polarities of up to 15 events is reshaped to a three dimensional tensor by concatenating the timestamps and polarities along the 15 entries. The resulting three dimensional tensor has two spatial dimensions and a channel dimension with 30 entries.

FLOP computation Tab. 1 shows the number of FLOPs to perform different operation in the network for standard networks and our method. The FLOPs

	Dense Layer	Sparse Layer
Convolution	$H_{\text{out}} W_{\text{out}} c_{\text{out}} (2k^2 c_{\text{in}} - 1)$	$N_r c_{\text{in}} (2c_{\text{out}} + 1)$
Max Pooling	$H_{\text{out}} W_{\text{out}} c_{\text{out}} k^2$	$N_a c_{\text{out}} k^2$
Fully Connected	$2c_{\text{in}} c_{\text{out}}$	$2c_{\text{in}} c_{\text{out}}$
ReLU	$H_{\text{out}} W_{\text{out}} c_{\text{in}}$	$N_a c_{\text{in}}$

Table 1: FLOPs computation at each layer. Here N_r are the number of rules at that layer and N_a are the number of active sites.

needed for a standard convolution is $H_{\text{out}} W_{\text{out}} c_{\text{out}} (2k^2 c_{\text{in}} - 1)$ excluding bias.

This is the result of performing $k^2 c_{\text{in}}$ multiplications and $k^2 c_{\text{in}} - 1$ additions for each pixel and each output channel resulting in the term found in the table. For our asynchronous sparse formulation we compute the number of operations by following Eqs. (M-8) and (M-9). Computing the differences in Eq. (M-8) results in $N_r c_{\text{in}}$ operations, where N_r are the number of rules at that layer. The convolution itself uses $N_r c_{\text{in}}$ multiplications and $N_r(c_{\text{in}} - 1)$ additions for each output channel, resulting in a total of $c_{\text{out}} N_r(2c_{\text{in}} - 1)$ operations. Finally, from Eq. (M-9) additional $N_r c_{\text{out}}$ operations need to be expended to add these increments to the previous state. In total, this results in $N_r c_{\text{in}}(2c_{\text{out}} + 1)$ operations.

1.4 Sensitivity on the Number of Events

Tab. 2 shows the computational complexity in MFLOPS and test accuracy on N-Caltech101[3] for both sparse and dense VGG13. The table shows that the test accuracy is maximized at 25'000 events for the sparse network and reaches a plateau for the dense network. At this number of events amount of computation in the sparse network is 46% lower than for the dense network. For this reason we selected 25'000 events for all our further experiments in the main manuscript.

	64		256		5000		25000		50000	
	Accuracy	MFLOP	Accuracy	MFLOP	Accuracy	MFLOP	Accuracy	MFLOP	Accuracy	MFLOP
Dense VGG13	0.257	1621.2	0.456	1621.2	0.745	1621.2	0.761	1621.2	0.766	1621.2
Sparse VGG13	0.247	224.2	0.435	381.0	0.734	697.5	0.745	884.2	0.730	959.2

Table 2: Computational complexity and test accuracy on N-Caltech101[3] for sparse and dense VGG13 and a varying number of events.

1.5 Qualitative Results on Object Detection

Here we show qualitative results of our method applied to the task of event-based object detection (Sec. M-4.2 and Tab. 4 in the main manuscript). Failure cases of our approach include very similar classes, such as "rooster" and "pigeon" in the third column. In the Gen1 Automotive dataset it can be seen that our approach works well for cars that are close and have a high relative motion. However, some cars are missed, especially if they have small relative speed and thereby only trigger few events (Fig. 1, bottom right).

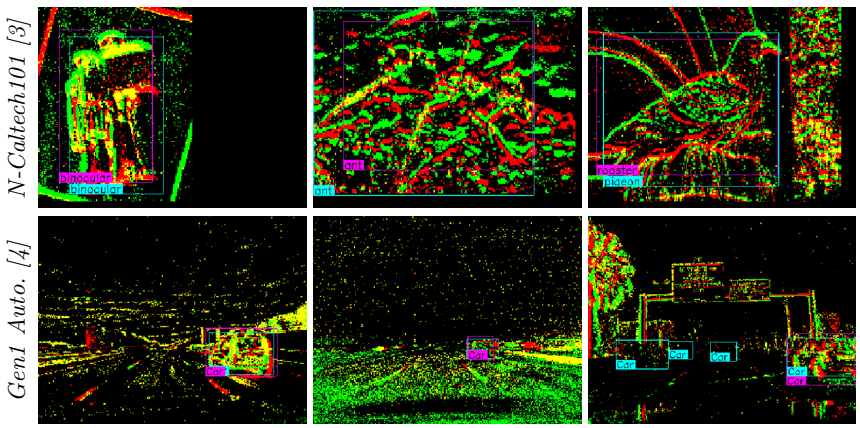

Fig. 1: Qualitative results of object detection (best viewed in color). Our predictions are shown in magenta, and labels in cyan. The first two columns present success cases, while the last column a failure case. On the first dataset, our method is mainly fooled by similar classes, such as "pigeon" and "rooster". In the second dataset, our approach detects cars generally well, but fails to detect the ones moving at similar speed due to the low event rate (bottom right).

Algorithm 1 Asynchronous Sparse Convolution at layer n

Update Active Sites
if the first layer ($n = 1$) **then**
- Update the active set $\mathcal{A}_t$ with all new active and new inactive sites
- Initialize the rulebook $\mathcal{R}_{\mathbf{k},0} = \emptyset$ and receptive field $\mathcal{F}_0 = \{\mathbf{u}_i\}_i$.

end if

Update rulebook and receptive field
- compute $\mathcal{R}_{\mathbf{k},n}$ using Eq. (M-7) with $\mathcal{F}_{n-1}$
- compute $\mathcal{F}_n$ using Eq. (M-6) with $\mathcal{F}_{n-1}$.

Layer update
for all $\mathbf{u}$ in $\mathcal{F}_n$ **do**
  if $\mathbf{u}$ remains an active site **then**
    - compute increment Δ_n using Eq. (M-8) with y^t_{n-1} and y^{t-1}_{n-1}
    - compute activation $\tilde{y}^t_n$ using Eq. (M-9) with Δ_n and $\tilde{y}^{t-1}_n$
  end if
  if $\mathbf{u}$ is newly active **then**
    - compute activation $\tilde{y}^t_n$ using Eq. (M-4) with y^t_{n-1}
  end if
  if $\mathbf{u}$ is newly inactive **then**
    - set activation $\tilde{y}^t_n$ to 0
  end if
  - compute y^t_n by applying a non-linearity as in Eq. (M-10)
end for

References

1. A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," in *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018, pp. 5419–5427.
2. S. Tulyakov, F. Fleuret, M. Kiefel, P. Gehler, and M. Hirsch, "Learning an event sequence embedding for dense event-based deep stereo," in *Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019.
3. G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Front. Neurosci.*, vol. 9, p. 437, 2015.
4. P. de Tournemire, D. Nitti, E. Perot, D. Migliore, and A. Sironi, "A large scale event-based detection dataset for automotive," *ArXiv*, vol. abs/2001.08499, 2020.