FARSE-CNN: Fully Asynchronous, Recurrent and Sparse Event-Based CNN Supplementary Material

Riccardo Santambrogio[1], Marco Cannici[2], and Matteo Matteucci[1]

[1] Politecnico di Milano, Milan, Italy
[2] University of Zurich, Zurich, Switzerland

1 Number of Active Sites

We now derive and prove the upper bound on the maximum number of sites a_n that can be affected, at layer n, by a single event at the input layer, as discussed in Sec. M-3.6[3]. Consider first a single layer, that convolves over a 1D space. Generalizing for 2D inputs, the derivations are applied to each dimension independently. The layer can either receive a single event or multiple events in *consecutive* sites. This is because in consecutive applications of convolutional operations, only nearby receptive fields will contain the same sites in input, as long as the convolution is not dilated (which we do not consider here). We can represent these blocks of inputs as small dense frames surrounded by inactive sites, or padding. Then, the number of active sites in output can be computed using the well known formula from convolution arithmetic:

$$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1 \tag{1}$$

where i, o are the input and output size respectively, k, s are the kernel size and stride and p is the amount of padding on each side. We want to account for all cells that would contain the input in their receptive field, in a potentially infinite line of cells. This is equivalent to assuming a padding $p = k - 1$, known as the full padding scheme. By substituting this value for the padding, Eq. (1) becomes

$$o = \left\lfloor \frac{i + k - 2}{s} \right\rfloor + 1. \tag{2}$$

When we move to considering multiple layers, the input size of layer n corresponds to the output size of the previous layer, $n - 1$. We can write this as

$$a_n = \left\lfloor \frac{a_{n-1} + k_n - 2}{s_n} \right\rfloor + 1 \tag{3}$$

using a_n for the number of active sites at layer n. The base case is $a_0 = 1$, for the single original event in the input layer. We can now solve this recurrence relation

[3] We indicate with M-* a reference in the main paper.

to obtain a closed-form upper bound for the number of active sites at layer n. To begin with, we upper bound the floor operation $\lfloor . \rfloor$ with its argument, to state

$$a_n \leq \frac{a_{n-1} + k_n - 2}{s_n} + 1. \tag{4}$$

Then, we proceed by substitution:

$$\begin{aligned}
a_n &\leq \frac{a_{n-1} + k_n - 2}{s_n} + 1 \\
&\leq \frac{1}{s_n}\left(\frac{a_{n-2} + k_{n-1} - 2}{s_{n-1}} + 1 + k_n - 2\right) + 1 \\
&\leq 1 + \frac{k_n - 1}{s_n} + \frac{a_{n-2} + k_{n-1} - 2}{s_n s_{n-1}} \\
&\leq 1 + \frac{k_n - 1}{s_n} + \frac{1}{s_n s_{n-1}}\left(\frac{a_{n-3} + k_{n-2} - 2}{s_{n-2}} + 1 + k_{n-1} - 2\right) \\
&\leq 1 + \frac{k_n - 1}{s_n} + \frac{k_{n-1} - 1}{s_n s_{n-1}} + \frac{a_{n-3} + k_{n-2} - 2}{s_n s_{n-1} s_{n-2}} \\
&\leq 1 + \frac{k_n - 1}{s_n} + \frac{k_{n-1} - 1}{s_n s_{n-1}} + \frac{k_{n-2} - 1}{s_n s_{n-1} s_{n-2}} + \frac{a_{n-4} + k_{n-3} - 2}{s_n s_{n-1} s_{n-2} s_{n-3}}.
\end{aligned}$$

At this point the pattern is evident, and we can guess our upper bound in closed-form to be the one of Eq. M-4, which we repeat here for the reader's convenience:

$$a_n \leq 1 + \sum_{i=1}^{n} \frac{k_i - 1}{\prod_{j=i}^{n} s_j}.$$

This is proved by induction, assuming that our guess holds for $n-1$:

$$\begin{aligned}
a_n &\leq \frac{a_{n-1} + k_n - 2}{s_n} + 1 \\
&\leq \frac{1}{s_n}\left(1 + \sum_{i=1}^{n-1} \frac{k_i - 1}{\prod_{j=i}^{n-1} s_j} + k_n - 2\right) + 1 \\
&= 1 + \frac{k_n - 1}{s_n} + \frac{1}{s_n}\left(\sum_{i=1}^{n-1} \frac{k_i - 1}{\prod_{j=i}^{n-1} s_j}\right) \\
&= 1 + \frac{k_n - 1}{s_n} + \sum_{i=1}^{n-1} \frac{k_i - 1}{\prod_{j=i}^{n} s_j} \\
&= 1 + \sum_{i=1}^{n} \frac{k_i - 1}{\prod_{j=i}^{n} s_j}
\end{aligned}$$

which confirms that our guess holds for n. The base case $a_0 = 1$ is also trivially proved, because the summation in Eq. M-4 is empty for $n < 1$.

Equation M-4 holds when all layers are regular convolutional layers, such as FARSE-Convolution and Sparse Pooling layers. When we use Submanifold FARSE-Convolution, instead, we know that the size of the output is the same

Table 1: FARSE-CNN Network Configurations. Submanifold FARSE-Convolution layers are followed by their kernel and hidden sizes. Sparse MaxPool layers are followed by their kernel size, and the stride for these layers is always equal to the kernel size. Temporal Dropout is followed by the window size, when specified. Highlighted rows indicate skip connections around the blocks. *The first Sparse MaxPool has kernel size 2 for the N-Cars and DVS128 Gesture datasets, and size 4 for the N-Caltech101 dataset. †The last fully connected layer outputs a flat tensor, but we reshape it into a 5×4 grid of bounding box predictions according to YOLO.

(a) Configurations for object and gesture recognition.

Sub FARSE-Conv 1×1, 16
Sub FARSE-Conv 3×3, 16
Sub FARSE-Conv 3×3, 16
Sparse MaxPool 2×2 / 4×4*
Temporal Dropout
Sub FARSE-Conv 3×3, 32
Sub FARSE-Conv 3×3, 32
Sparse MaxPool 2×2
Temporal Dropout
Sub FARSE-Conv 3×3, 64
Sub FARSE-Conv 3×3, 64
Sparse MaxPool 2×2
Temporal Dropout
Sub FARSE-Conv 3×3, 128
Sub FARSE-Conv 3×3, 128
Sparse MaxPool 2×2
Temporal Dropout
Sub FARSE-Conv 3×3, 256
Sub FARSE-Conv 3×3, 256
Sparse AvgPool to 1×1
LSTM, 128
FC + softmax

(b) YOLO configuration for object detection.

Layer	Output Size
Sub FARSE-Conv 1x1, 16	304×240
Sub FARSE-Conv 3×3, 16	304×240
Sparse MaxPool 4×4	76×60
Temporal Dropout 4	76×60
Sub FARSE-Conv 3×3, 32	76×60
Sparse MaxPool 2×2	38×30
Temporal Dropout 4	38×30
Sub FARSE-Conv 3×3, 64	38×30
Sparse MaxPool 2×2	19×15
Temporal Dropout 4	19×15
Sub FARSE-Conv 3×3, 128	19×15
Sparse MaxPool 2×2	10×8
Temporal Dropout 4	10×8
Sub FARSE-Conv 3×3, 256	10×8
Sparse MaxPool 2×2	5×4
FARSE-Conv 5×4, 256	1
FC 240	5×4†

as the size of the input, *i.e.*, $a_n = a_{n-1}$, and the layer does not influence the number of inputs for layers that follow. Therefore, we simply consider $k_i = 1$ if layer i is a Submanifold FARSE-Convolution, while the stride s_i is already restricted to 1 by design.

2 Networks Details

In Tab. 1a we show the detailed configurations of the networks used for our experiments on object recognition (Section M-4.2) and gesture recognition (Section M-4.3). We first apply a 1×1 Submanifold FARSE-Convolution and then 5 blocks of 3×3 Submanifold FARSE-Convolutions alternated by compression blocks. In the compression blocks we have a Sparse Max Pooling followed

by a Temporal Dropout, so that the network learns to aggregate event features in each spatio-temporal neighborhood. Around the blocks of Submanifold FARSE-Convolutions we use skip connections, by summing together the event features (this is possible because Submanifold FARSE-Convolutions alone do not change the spatio-temporal structure of the event stream). For the N-Caltech101 dataset, we change the first Sparse Max Pooling layer to a kernel 4×4, performing a stronger spatial compression due to the larger resolution of this dataset.

For experiments on object detection (Section M-4.4), we build a FARSE-CNN with a fully-connected YOLO output layer, as shown in Tab. 1b. In this case, the blocks of two Submanifold FARSE-Convolutions are replaced with single layers, leading to lower computational complexity as discussed in Sec. M-4.4. Skip connections are still used. After the stream is brought down to size 5×4, a cell of FARSE-Convolution is applied in a fully-connected way (*i.e.*, kernel size and stride are both 5×4), and its output is passed to a linear fully-connected layer with size 240. The final output is reshaped again into a 5×4 grid, where each cell predicts two bounding boxes ($2 * 5$ values) and two logits for the class probabilities.

3 Networks Complexity

In Sec. M-4 we report the MFLOP/ev on each dataset by counting the effective number of inputs received and aggregations performed at each layer. However, the upper bounds provided in Sec. M-3.6 give us the complexity of any layer that is reached by input events. To analytically compute the complexity of an entire network, independently of the data it is used on, we must only take care of the effect of Temporal Dropouts. For networks that use Submanifold FARSE-Convolutions, like in our configurations, this can be done by considering the *amortized* cost of all layers. A Temporal Dropout with window size l, after an initialization phase (*i.e.*, if the number of inputs received by the layer is much larger than the size of its grid), only propagates $1/l$ of its inputs forward. Hence, the amortization factor of any layer is equal to the product of the inverse window size $1/l$ of all Temporal Dropouts preceding it. For instance, if a certain layer is preceded in the network by two Temporal Dropouts at n_1 and n_2, its amortization factor will be equal to $1/(l_{n_1} \cdot l_{n_2})$. Note that this amortized analysis cannot be applied to networks that use normal FARSE-Convolutions together with Temporal Dropouts. This is because FARSE-Conv layers generate blocks of contiguous events, as explained in Sec. M-3.3, some of which are dropped, while others might be propagated due to the independent counters of Temporal Dropout cells. Therefore, computing an upper bound on the complexity of these networks would require a more sophisticated amortized analysis.

Using this reasoning, we obtain for our configurations the values shown in Tab. 2. As can be seen, the empirically measured MFLOP/ev are on some datasets higher than the theoretical ones. This is because the ratio of events that are propagated through the network is higher during an initial phase, and only converges to the one considered in the theoretical amortized analysis as the

Table 2: Theoretical MFLOP/ev of FARSE-CNN networks. The theoretical complexity depends only on the configuration of each network, although empirically we observe small variations based on the dataset of application.

Network	Dataset	MFLOP/ev	
		Theoretical	Empirical
TD window 2	DVS128 Gesture	2.042	1.905
	N-Cars		2.056
TD window 4	N-Cars	0.338	0.495
	N-Caltech101		0.339
YOLO	Gen1 Automotive	0.155	0.137

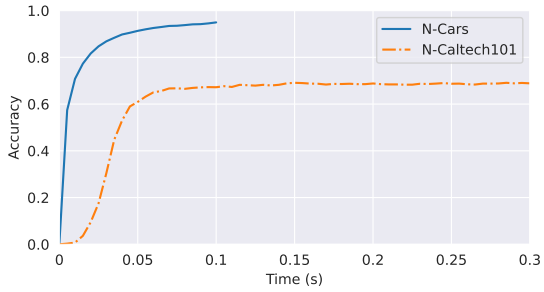

Fig. 1: Accuracy over time on object recognition datasets. The best performing networks on each dataset are used.

network receives more input events. Hence, the computed value represents an asymptotic upper bound on the MFLOP/ev of the network. Indeed, in Tab. 2, the empirical complexity remains below the theoretical one for datasets containing longer sequences with more events, such as DVS128 Gesture and Gen1 Automotive.

4 Accuracy Over Time

FARSE-CNN outputs a stream of predictions driven by incoming events, so we can inspect how the accuracy of the model changes over time. This is visualized in Fig. 1 for object recognition datasets. The plot shows that the model's output has a transient period before converging to a stable answer. On the shorter samples of N-Cars the accuracy steadily increases until the end of the sequences, while on N-Caltech101 it reaches a plateau. In both cases, the model reaches significant accuracy already by the 50 ms mark, long before the total length of samples.

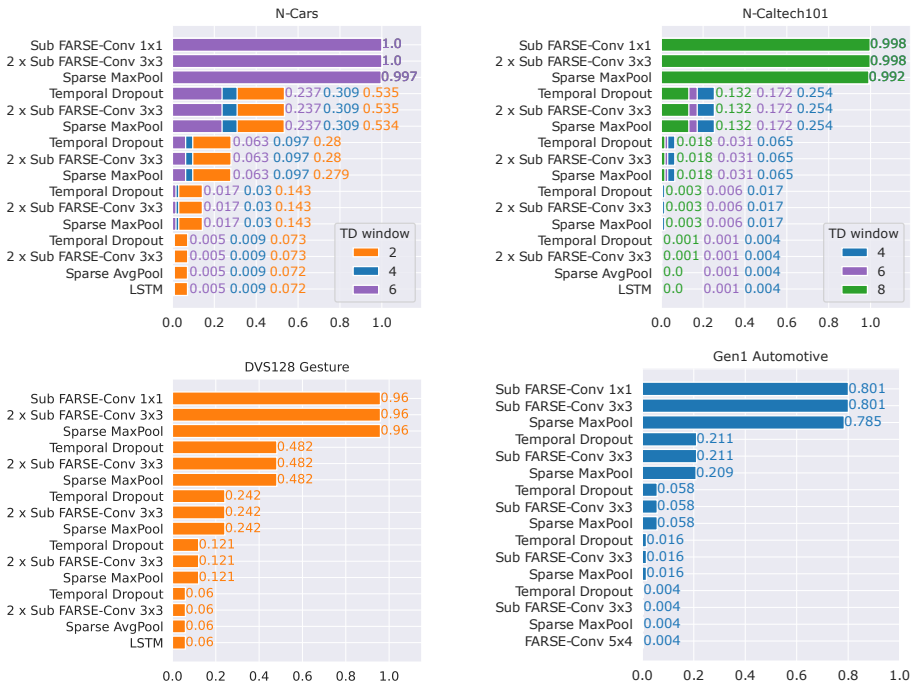

Fig. 2: Internal events triggered at each layer, expressed as ratio to the number of input events.

5 Analysis of Temporal Dropout and Internal Events

In Fig. 2 we show the ratio of events that are triggered internally by each layer in our networks, on the different datasets. While a small amount of compression is performed by Sparse Pooling operations, the largest impact on the number of internal events is achieved by the temporal compression of Temporal Dropout. When Temporal Dropout uses a window size of 4, less than 1% of the input events received by the network are propagated through to the output.

It can be noted that, in some cases, the ratio of events falls below 1.0 already at the first layers, before any compression module is applied. This is due to the presence of multiple events sharing the same timestamp and coordinates in the original data. Our FARSE-CNN processes them as intended, by aggregating the simultaneous inputs and triggering a single output event, leading to the observed reduction.

6 Noise Sensitivity

Since event cameras can be noisy, generating lots of unwanted events, the concern is raised of how FARSE-CNN behaves under heavy noise. It should be noted that

Table 3: Accuracy at different levels of noise.

Noise level	0%	1%	5%	10%	20%	30%
N-Caltech101	0.687	0.687	0.685	0.678	0.650	0.599
N-Cars	0.949	0.950	0.934	0.915	0.889	0.858
N-Cars (trained with noise)	0.946	0.950	0.947	0.941	0.934	0.917

all data used in our experiments was recorded using real event cameras, therefore they already consider real-world noisy data. To further validate noise sensitivity, we test FARSE-CNN on N-Cars and N-Caltech101 by adding different amounts of noisy events[4], computed as percentages of the average number of events in a sample, in Tab. 3.

The networks trained on the original data show good robustness to reasonable levels of noise, which is improved on N-Cars if we re-train the network with noise sampled in the same maximum range. In terms of efficiency, the total computation increases by the number of added events, but the MFLOP/ev remains the same: in the maximum noise case (30%) we measure 2.063 MFLOP/ev on N-Cars and 0.335 MFLOP/ev on N-Caltech101, which is virtually unaffected compared to the noise-free case (see Tab. 2).

[4] `https://tonic.readthedocs.io/en/latest/autoapi/tonic/functional/uniform_noise`

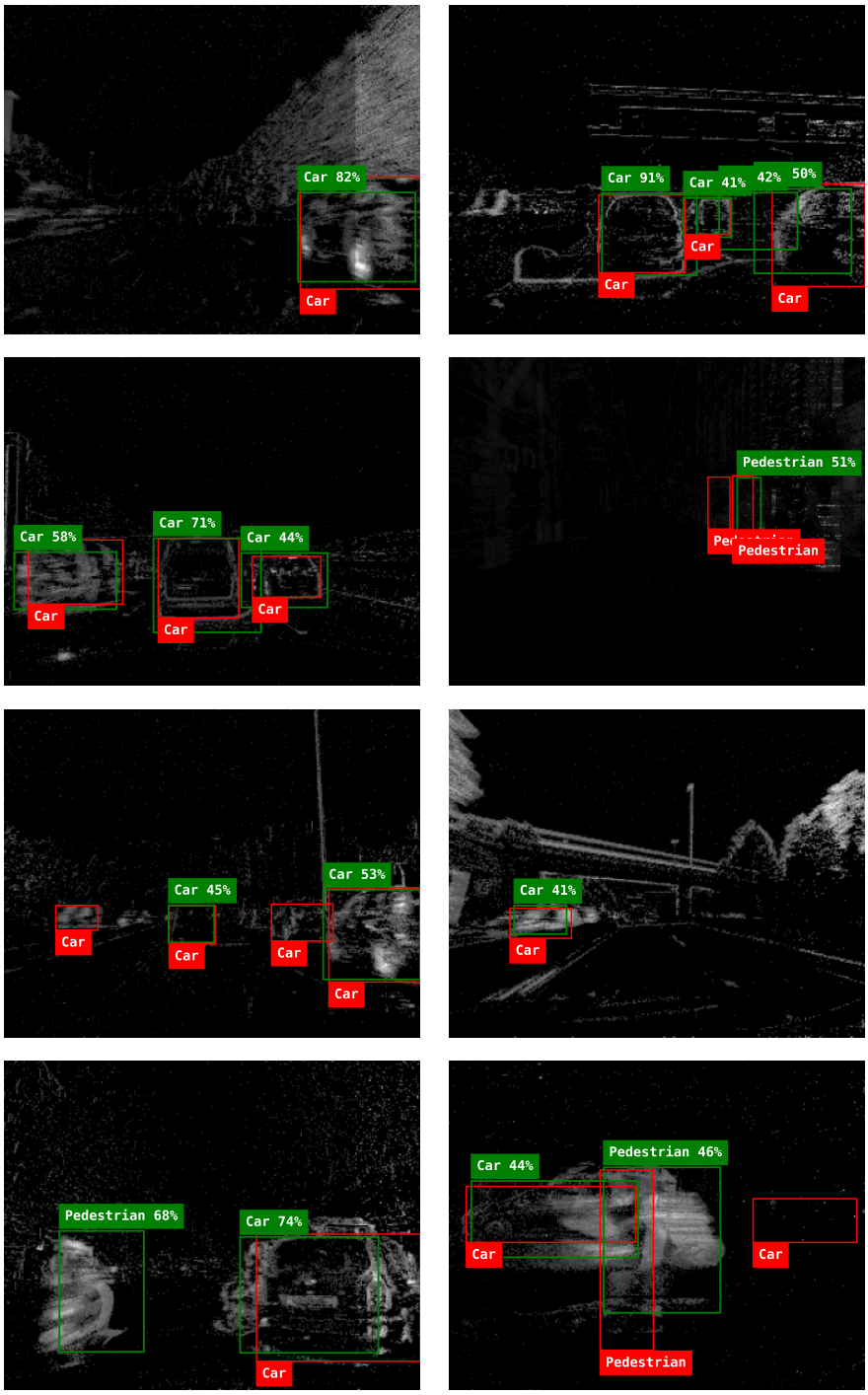

Fig. 3: Examples of FARSE-CNN detections. Predictions are shown as green boxes, while ground truth boxes are in red. We show both correct predictions and mistakes of our model.