Supplementary Material — AEGNN: Asynchronous Event-based Graph Neural Networks

Simon Schaefer* Daniel Gehrig* Davide Scaramuzza

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

1. Appendix

Here we report additional information to support the main manuscript. In what follows, we will refer to figures, tables, sections, and equations from the manuscript by prepending "M-". We start by providing a detailed network overview in Sec.1.1. We then show a derivation for the number of FLOPS required to compute the Spline Convolution in [2] for a single node in Sec.1.2. We finally list the licenses of the datasets used in this submission in Sec. 1.3.

1.1. Network Details

We use two network architectures in this work, one for object recognition (Sec. M-5.2) and one for object detection (Sec. M-5.3). Both networks consist of convolutional blocks, each containing a SplineConv [2], defined by the number of output channels M_{out} and kernel size k, an ELU activation function, and a batch norm. As shown in Figure M-2, max graph pooling layers after the fifth and seventh convolution, as well as skip connections after the fourth and fifth convolution, are used. Also, a fully connected layer maps the extracted feature maps to the network outputs. The recognition network has convolutions with kernel size $k = 2$ and output channels $M^i_{out} = (1, 8, 16, 16, 16, 32, 32, 32)$. The convolutions in the detection network have a much larger kernel size $k = 8$ and more output channels $M^i_{out} = (1, 16, 32, 32, 32, 128, 128, 128)$.

1.2. Spline Convolutions Complexity

In this work, we make heavy use of Spline Convolutions [2]. Compared to standard GNN layers which only aggregate features over layers, Spline Convolutions also take into account the spatial arrangement of these neighbors and thus produce richer features. Here we will give a summary of spline convolutions and refer the reader to [2] for more details. Given nodes i with features $\mathbf{f}(i)$ we define convolution kernels g_n, with $n = 1, ..., M_{\text{out}}$ the index of the output feature. They act as

$$(g_n * f)(i) = \frac{1}{|N(i)|} \sum_{l=1}^{M_{in}} \sum_{j \in N(i)} f_l(j) g_{n,l}(u(i,j)) \quad (1)$$

Here $f_l(j)$ is the input feature with index j, $N(i)$ counts the number of neighbors of node i, and $u(i,j)$ are pseudo coordinates. These are defined as the normalized distance vector between nodes i and j.

The function g is expanded as

$$g_{n,l}(u) = \sum_{p \in P} w_{p,l,n} B^m_p(u) \quad (2)$$

Here P denotes an index set, which is a regular grid in 3 dimensions. It has two elements in each direction, resulting in $2^3 = 8$ elements (tuples). For each coordinate tuple a learnable weight $w_{p,l,n}$ is stored and multiplied by a B-Spline basis $B^m_p(u)$ in three dimensions. Each B-Spline basis is computed by forming the product of three splines as

$$B^m_p(u) = \prod_{s=1}^{3} N^m_{p_s,s}(u_s), \quad (3)$$

i.e. one for each dimension. Here m is the degree of the B-Spline, and in this work we use $m = 3$. Each function $N^m_{p_s,s}(u_s)$ can thus be written $N^m_{p_s,s}(u_s) = \sum_{j=0}^{m-1} a_j u_s^j$.

1.2.1 FLOPS Computation

Here we count the FLOPS necessary. In what follows we define $N_i = |N(i)|$ and $N_p = |P|$ and will proceed in a series of steps. To evaluate Eq. 1 we first compute u for all neighbors and dimensions, resulting in:

$$FLOPS(u(i,j)) = N_i d \quad (4)$$

then we compute the FLOPS to evaluate $B^m_p(u)$ as

$$FLOPS(B^m_p(u)) = 2mN_i d + N_i(d-1) \quad (5)$$

*these authors contributed equally

For the first term we make use of Horner's method [3], which states the optimal number of additions and multiplications for a polynomial of degree m as $2m$. For the second term we count the FLOPS to compute the product. Each operation needs to be repeated for each neighbor. Next we compute $g_{n,l}(u)$ as the sum of products over elements of P, input features and output features.

$$FLOPS(g_{n,l}(u)) = (2N_p - 1)M_{\text{out}}M_{\text{in}}N_i \qquad (6)$$

Finally we aggregate these terms, first over neighbors and then over input features

$$FLOPS\left(\sum_{l=1}^{M_{\text{in}}}\sum_{j\in N(i)} f_l(j)g_{n,l}(u(i,j)\right) = (2N_i - 1)M_{\text{out}}M_{\text{in}} + (M_{\text{in}} - 1)M_{\text{out}}$$

Where the first term counts products and summation over neighbors, and the second counts summation over input features. Finally, we divide all output features by N_i, adding additional M_{out} FLOPS. We thus have

$$\begin{aligned} C_{\text{tot}} =& N_i d + 2mN_i d + N_i(d-1) \\ &+ (2N_p - 1)M_{out}M_{in}N_i \\ &+ (2N_i - 1)M_{out}M_{in} \\ &+ (M_{in} - 1)M_{out} + M_{out} \\ =& N_i M_{out}M_{in}(1 + 2N_p) + N_i(2d + 2md - 1). \end{aligned}$$

1.3. Licenses

We use the Prophesee Gen1 dataset [1] under the "PROPHESEE GEN1 AUTOMOTIVE DETECTION DATASET LICENSE TERMS AND CONDITIONS" found at the URL https://www.prophesee.ai/2020/01/24/prophesee-gen1-automotive-detection-dataset/. N-Caltech101 [4] is used under the "Creative Commons Attribution 4.0 license" and downloaded at https://www.garrickorchard.com/datasets/n-caltech101. Finally, the N-Cars dataset [5] is also released by Prophesee under https://www.prophesee.ai/2018/03/13/dataset-n-cars/.

References

[1] Pierre de Tournemire, Davide Nitti, Etienne Perot, Davide Migliore, and Amos Sironi. A large scale event-based detection dataset for automotive. *arXiv e-prints*, abs/2001.08499, 2020. 2

[2] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. *CVPR*, 2018. 1

[3] W.G. Horner. In *Philos. Transact. Roy. Soc. London*, 1819. 2

[4] Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Front. Neurosci.*, 9:437, 2015. 2

[5] Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. HATS: Histograms of averaged time surfaces for robust event-based object classification. In *CVPR*, pages 1731–1740, 2018. 2