Supplementary materials G2N2: Lightweight Event Stream Classification with GRU Graph Neural Networks

Thomas Mesquida[1]
thomas.mesquida@cea.fr
Manon Dampfhoffer[2]
manon.dampfhoffer@cea.fr
Thomas Dalgaty[1]
thomas.dalgaty@cea.fr
Pascal Vivet[1]
pascal.vivet@cea.fr
Amos Sironi[3]
asironI@prophesee.ai
Christoph Posch[3]
cposch@prophesee.ai

[1] Univ. Grenoble Alpes
CEA-List
Grenoble, France
[2] Univ. Grenoble Alpes
CEA, CNRS, Grenoble INP,
INAC-Spintec
Grenoble, France
[3] Prophesee
Paris, France

This appendix aims to provide a comprehensive understanding of the key concepts discussed in the main paper. In Section 1, we exhibit the drawbacks associated with existing asynchronous methods for constructing and computing with Event-Graphs (EGs) Section 2 expands on the limitations of the Feed Forward Task Head and its impact on performance. Further experiments focusing on the temporal aspects of EGs and the utilization of GRU are presented in Sections 3 and 4, respectively. Lastly, in Section 5, we offer additional details regarding the lightweight CNN baselines proposed in this study.

1 Asynchronous Event-based Graph Neural Networks

Algorithm 1 explains the method used in [2] to construct Event Graph (EG) and compute Event Graph Neural Network (EGNN) asynchronously using a Fully-Spherical update (as opposed to the Hemi-Spherical update [1] described in the main paper). When a new event arrives, its direct neighborhood, i.e. all nodes within the defined radii, has to be processed for potential update. Indeed, due to future-to-past connections in fully-spherical update, this new event can share data to an already existing node, updating its adjacency table and features.

This difference in connectivity is shown in Fig. 1, depicting the state of the EG after all of the events have been received. The top row represents edges the considered node will take data from, or input edges, i.e. the result of the search algorithm. Bottom row represents all

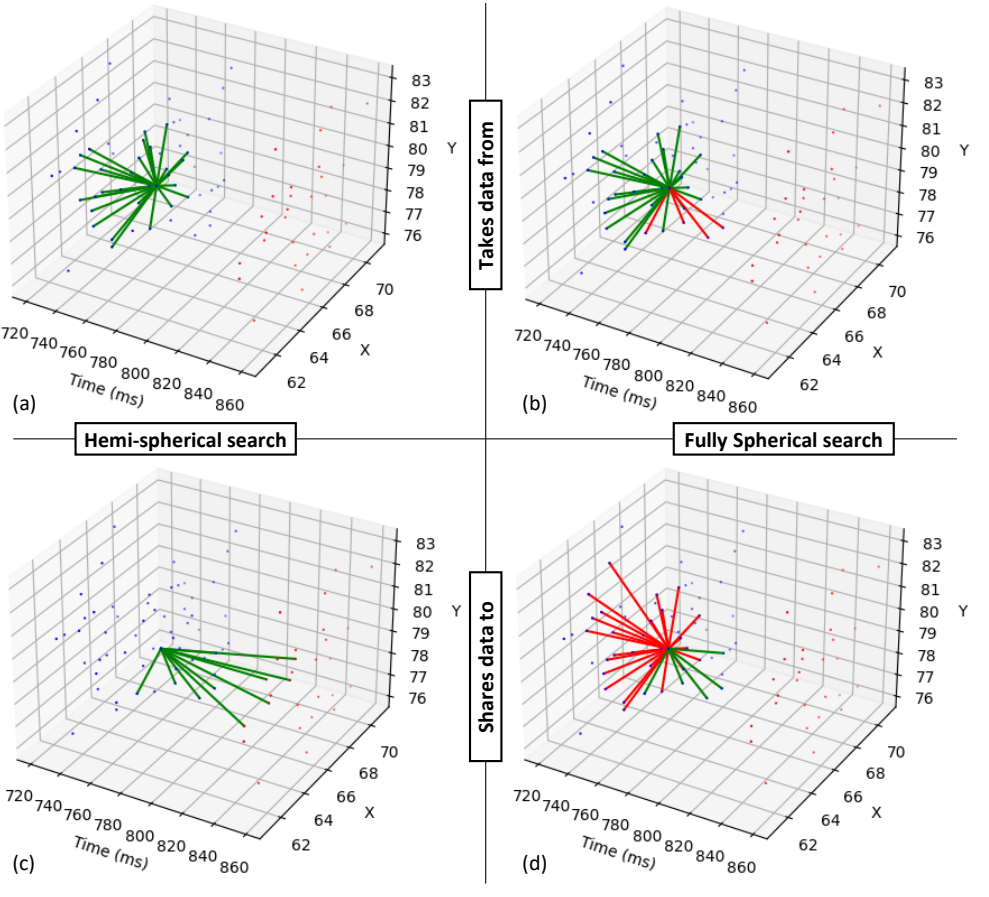

Figure 1: Edges created around a node. Blue and red dots represent negative and positive events respectively. Green and red edges represent past-to-future and future-to-past edges respectively. (a) and (b) (resp. (c) and (d)) represent input (resp. output) edges to the considered node. (a) and (c) use Hemi-Spherical search [1]. (b) and (d) use Fully-Spherical Search.

Algorithm 1 Sparse Fully-Spherical update.

Input: New event $ev = (x, y, t, p)$, Event Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, radii r_{xy} and r_t, L layers EGNN, Task Head
Output: Updated Event Graph, Features, Outputs

Add ev to the vertices $\mathcal{V}$
for Event ev_i within radii from ev **do**
    Update $\mathcal{E}$ for event ev_i with KNN ▷ ev can replace vertex previously in KNN
end for
$\mathcal{G}$ **has been updated**

for Layer in EGNN of index i **do**
    for Event ev_j within i hops of ev in $\mathcal{G}$ **do**
        Update Features for ev_j at layer i
    end for
end for
Features have been updated

for Event ev_i within L hops of ev in $\mathcal{G}$ **do**
    if Features of ev_i have been updated **then**
        Compute new Output for ev_i with Task Head
    end if
end for
Outputs have been updated

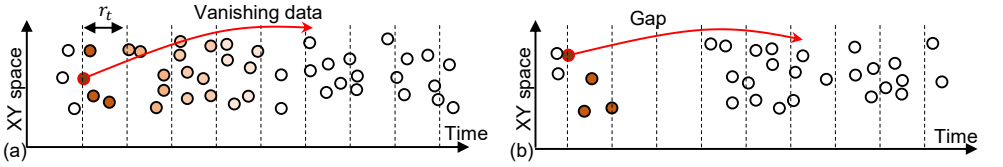

Figure 2: Illustration of vanishing features in EGNN. A feature is computed at the node circled in red. For a 5 layers EGNN, its data vanishes in 5 times the temporal search radius (left) or with gaps in the graph (right). Color intensity indicates at which layer of the EGNN the feature can be perceived, the lighter the further.

of the edges it will share data to, or output edges, i.e. the result of the search algorithm for other nodes in the vicinity that include the considered node.

Green and red edges are past-to-future and future-to-past edges respectively. At the time the event arrived, adding this new node to the EG creates at least the red output edges (bottom) and requires an update of the impacted nodes. Some of these can be removed later due to new events being closer to the receiving nodes, inducing further updates. Input edges (top) to this new node are at this point the green input edges. Later arrival of close events create the red input edges, inducing updates for the considered node.

Moreover, this reasoning can be extended past this single neighborhood search. Indeed, as a new event arrives, it has the potential to change the output features of the first layers of the EGNN for all existing nodes within its direct neighborhood. Then, these nodes are effectively sharing new features at the input of the second EGNN layer. Thus, the computation for the nodes in their adjacency table have to be updated for layer 2. This ensemble of nodes can be defined as second degree neighbor from the new event, i.e. that can be reached with 2 hops in the graph (passing from one vertex to another through an edge).

Extending this to a N-layer EGNN, all nodes within N hops from the new event will have to update some of its features. This means that node features become update free and can be computed safely only after N times the temporal search radius, guarantying it cannot be reached in N hops. Therefore, using Fully-Spherical search to create an EG induces additional computation (as new events can update past results) and adds N times the temporal search radius in latency. With our 5-layer EGNN and 100 ms temporal search radius, this corresponds to 500 ms and prohibits the use of such solutions for real time applications.

2 Vanishing features in EGNN

In our evaluation of different task heads for integration with the EGNN feature extractor, we found that the Feed Forward Task Head (FFTH) performed poorly compared to GRU-based task heads. The limitation of FFTH lies in its ability to compute output scores only on a per-node basis, relying on the EGNN feature extractor to capture temporal information. However, EGNN inherently struggles to analyze patterns over extended periods due to the vanishing nature of its computed features, as depicted in Figure 2.

As explained in Section 1, an event can only influence features of nodes in N hops from itself at the N^{th} layer of the EGNN. This is the first source of vanishing features in the EG. We used 100 ms time search radius, meaning for our 5-layer EGNN that inputs features cannot

Neighbor Search	Search radii	Search Volume	Latency	Acc
Fully-spherical	(10 pix, 50 ms)	V	250 ms	68.3 %
Fully-spherical	(10 pix, 100 ms)	2V	500 ms	68.0 %
HUG	(10 pix, 100 ms)	V	0 ms	**69.4%**
HUG	(10 pix, 200 ms)	2V	0 ms	68.1 %
HUG	(10 pix, 400 ms)	4V	0 ms	67.3 %

Table 1: Comparison of EG building parameters. Hemi-spherical Update Graph (HUG) [1] vs Fully-Spherical Update. GRU is processed at 75 Hz. Latency represents the time between creation and output feature computation for an event.

be kept more than 500 ms in the EG. Most samples are longer than that and early features are not kept until the end. Features can also vanish due to gaps in the EG, i.e. periods of time greater than temporal search radius with no event that prohibit edge creation. Therefore, EGNN must be combined with a recurrent task head in order to correctly analyze patterns over a longer period of time.

3 Time search radius

We chose to exhibit results for search radii of 10 pixels and 100 ms, searching in the past only. This represents a search volume V. For a fair comparison, the fully-spherical search was performed using half the time search radius, 50 ms. Doing so, we obtained the same search radius V .

Table 1 shows additional results for graph building with different time search radius. Extending the volume past current values do not increase the accuracy.

4 GRU rate

Table 2 depicts accuracy for DVS-Lip at different GRU rates. 75 Hz achieves the best accuracy, in line with conv-based topologies in [3]. Higher rates induce additional computation while performing worse. Although performance degradation with rate is not as important as in conv-based topologies [3]. While CNN feature extractor follow the same rate, EGNN inference is independant of said rate. Thus, GRU rate can be changed without affecting feature extraction. Conv-based approaches have a hard limit on rates: setting rates too high will result in poor performance as there is not enough activity for the CNN feature extractor to grasp patterns [3].

5 CNN-based topologies

CNN-based topologies all use ReLU activation function. Batch Normalization layers are also added after each convolutional layer. These layers's parameters are fused into previous layer's weight and bias at inference time and thus do not add computational cost at inference time. The Instance Normalization layer did not bring further gains combined with existing Batch Normalization layers and was removed from the Task Heads for conv-based features extractors.

GRU rate	Acc
30 Hz	66.1 %
75 Hz	**69.4** %
90 Hz	66.8 %
120 Hz	67.4 %
150 Hz	67.3 %
180 Hz	68.0 %

Table 2: Comparison of GRU rate. Maskout parameters are adapted so the maximum duration stays the same.

References

[1] Thomas Dalgaty, Thomas Mesquida, Damien Joubert, Amos Sironi, Pascal Vivet, and Christoph Posch. Hugnet: Hemi-spherical update graph neural network applied to low-latency event-based optical flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 3952–3961, June 2023.

[2] Simon Schaefer, Daniel Gehrig, and Davide Scaramuzza. Aegnn: Asynchronous event-based graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12371–12381, 2022.

[3] Ganchao Tan, Yang Wang, Han Han, Yang Cao, Feng Wu, and Zheng-Jun Zha. Multi-grained spatio-temporal features perceived network for event-based lip-reading. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20062–20071, 2022. doi: 10.1109/CVPR52688.2022.01946.