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

Abstract

Event camera pixels efficiently encode visual information through triggered events, offering advantages in temporal detail, dynamic range, and data reduction. However, the optimal machine learning method for leveraging these characteristics remains unclear. Existing approaches often convert events into 2D frames, losing crucial time-domain information. A promising alternative is event-graph neural networks, but they suffer from computational intensity and limited temporal dependencies. As a solution, we propose to combine the recently proposed lightweight event-graph neural network HUGNet with gated recurrent units to model temporal dependencies between the features extracted by HUGNet. We benchmark our model against other event-graph and convolutional neural network based approaches on the challenging DVS-Lip dataset (spoken word classification). We find that not only does our method outperform state of the art approaches for similar model sizes, but that, relative to the convolutional models, the number of calculation operations per second was reduced by 81%.

Furthermore, we introduce a new event-data augmentation technique that boosts by up to 7.4% the performance of both event-graph and convolutional neural networks on this task.

1 Introduction

Industrial event-cameras [10, 23] offer a new approach to artificial vision by capturing spatiotemporal activity at microseconds time scale. Unlike conventional frame-based cameras, which periodically record absolute light intensity, event-camera pixels generate binary flags

(referred to as events) upon the detection of relative light intensity changes in a purely event-driven and asynchronous fashion.

The state of the art solution for treating such data has typically been to convert a stream of event-data back into 2D frames (called dense or event-frames) which serve as the input to convolutional neural networks[11, 17, 19]. While this works well in practice, there are some potential pitfalls. For instance, not only are the precise temporal relationships between event-generating bodies distorted, but the opportunity for sparse and event-driven real-time computation is largely lost.

An emerging alternative solution are **E**vent-**G**raph **N**eural **N**etworks (EGNNs) [3, 6, 21, 26]. Rather than compacting events into frames, EGNNs build a graph in which each event is a vertex and edges are formed between vertices based on a search within a Euclidean ellipsoidal volume around each vertex. Edges can also be used to preserve the original spatiotemporal structure of the input event-data, which can thereafter be leveraged in graph convolution [8]. Early results have shown that EGNNs can outperform CNNs based on event-frames over a range of tasks, despite a greatly reduced model footprint [3, 21, 26]. However, due largely to the computationally expensive graph building procedure, efficient and real-time operation remains problematic. Furthermore, while **E**vent-**G**raphs (EGs) naturally capture short range temporal dependencies via edges formed to events generated in the past, these dependencies will be limited to times on the order of the temporal search radius - potentially only some tens or hundreds of milliseconds.

In this article, to solve these two problems, we combine the recently proposed hemispherical update graph neural network method (i.e., HUGNet) [6] with gated recurrent units (GRUs) [5] in GRU Graph Neural Network (G2N2). The GRU layers analyse temporal patterns in the features extracted by the Graph layers. Relative to other computationally costly EG methods, HUGNet prevents future nodes to influence past results, enabling lightweight feature extractor. This limitation in information sharing increased Optical Flow accuracy [6] and we extend this result to classification. We apply our model to event stream classification using the DVS-Lip dataset [28] and compare ourselves to other fully-spherical EGNNs as well as CNNs based on event-frames. The specific contributions of this work are as follows:

- We combine EG neural network feature extractors with gated recurrent units in G2N2, enabling lightweight modelling of temporal dependencies in event-data.
- We show that EG neural networks are better feature extractors than convolutional neural networks for event-data.
- We observe that hemi-spherical EG updates outperform fully-spherical updates.
- We propose a mask-based data augmentation method called Maskout that significantly boosts performance for both EG and convolutional models.

2 Related Work

Event-camera data have often been treated by using convolutional neural network architectures - developed initially for application to traditional static dense frames [11, 12, 15, 19, 24, 34]. In order to convert event-data into frames, there are a number of techniques. Most often this is achieved by counting the number of generated events at each pixel within a temporal window of some tens or hundreds of milliseconds [11, 17]. In fact, empirical results have found that CNNs using event-frames could outperform CNNs using standard frames [19] on

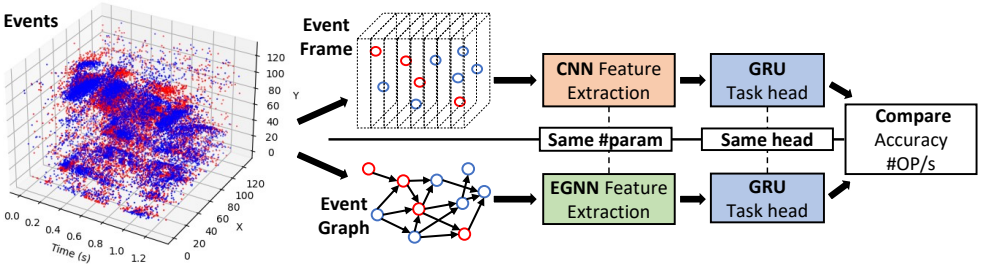

Figure 1: EGNN and CNN feature extractor are compared in term of accuracy and number of operations per second to process DVS Lip dataset. The comparison is done using the same training hyperparameters, task heads and number of parameters in feature extractors.

the same task. To avoid completely discarding temporal correlations, some works have proposed the use of time surfaces [16, 27] where pixel intensities correspond to, for example, the time since the last event was generated at each pixel. Other works also combine time surfaces with event counting [34] or use recurrent neural networks to convert event streams to frames [4]. Event networks [7] processed asynchronously could keep temporal details but at the expense of important memory overhead. Spiking neural networks have been applied to event-data [1, 32]. Although they are naturally compatible with asynchronous and event-based data, spiking models generally fail to attain the performance of CNNs on the same task [2, 22] due to spiking activation and requiring backpropagation through time.

In recent years, graph neural networks [8, 13, 14, 29] have been utilized to process event-camera data by constructing graphs to which graph convolutions are applied. Events are typically represented by two spatial coordinates (i.e., pixel address) and a timestamp. The timestamp is scaled through multiplication with a constant and a KD-tree data structure is then constructed [33]. By performing a K-nearest neighbour search on the resulting tree, each event is then connected to those in its vicinity by an edge [8, 21, 26] - thereby creating an EG. The use of EGs enables the retention of the fine spatiotemporal structure of the event-data in the graph edges - for example, as normalised vectors describing local spatiotemporal differences [8]. Despite promising early results, the existing EG building paradigms incur a latency which is too large to be compatible with real-time operation. Current methods construct graphs using a *fully-spherical* search where edges are formed from past to future events and also from future to past events. This approach has two significant drawbacks. Firstly, event-level predictions cannot be made instantaneously. Newly arrived events may be updated by an event arriving within a time equal to the temporal search radius multiplied by the number of graph layers, which may prove critical in time-sensitive applications. Secondly, the node embeddings of previously arrived events within the same temporal window are subject to change, and graph convolutions may need to be applied several times, which can be particularly problematic for high-resolution event cameras that generate millions of events per second [10]. To address these issues, new *hemi-spherical* methods have been proposed [6] which may allow for the processing of event streams in real-time. Events are limited to form edges from past events to events that will be generated in the future only. This eliminates the need to reapply graph convolutions after each new event arrival and permits instant event-level predictions and faster processing of event streams.

Event-based lip reading was recently proposed as a challenging neuromorphic benchmark task [28] whereby an event-camera records the activity of speakers mouths as they utter words - the objective being to classify the word spoken. Existing solutions range from

the application of several branches of 2D and 3D convolutions applied to event-data in respective voxel grids of differing temporal granularity [28] to using reservoir computing as feature extractors [31].

3 Method

3.1 Building Event Graphs

The first step when using EGNNs is to build, or update, an existing EG from an event stream. Event streams $\mathcal{S}$ can be described as a series of events e_i, each of which arrives at a time t_i with pixel coordinates (x_i, y_i) and a polarity p_i such that $\mathcal{S} = (e_i) = (x_i, y_i, t_i, p_i)$. A widely used method in the state of the art is to define a limited number of nearest neighbours N by performing a Euclidean search around each event within a temporal radius r_t and a pixel radius r_{xy}. Edges are defined between these N nearest neighbours in order to create an Event-Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ composed of Vertices $\mathcal{V}$ and Edges $\mathcal{E}$.

Algorithm 1 Sparse hemi-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 $\mathcal{V}$
Update $\mathcal{E}$ for event ev with KNN ▷ New vertex cannot modify past graph by design
$\mathcal{G}$ **has been updated**

Compute Features for ev with EGNN ▷ No edge update means no feature update needed
Features have been updated

Compute Output for ev with Task Head ▷ No feature update means no output update
Outputs have been updated

The resulting graph $\mathcal{G}$ is processed using graph convolutions in EGNNs in order to perform a desired task. It should be noted that, during GPU-based training, EG creation is generally not performed as described above, in an on the fly fashion, as performing the above steps event-by-event is highly time consuming. While methods have been proposed to permit EGs to be updated sparsely in an asynchronous fashion [26], the state of a particular vertex can only be guaranteed after a time equal to the number of graph convolution layers multiplied by the temporal search radius has elapsed (see Supplementary materials). Furthermore, the vertex features (also referred to as node embeddings) may be modified several times as new events arrive with repeated graph convolutions. This not only imposes a lower-bound on the prediction latency, but also implies a large amount of computation. EG creation may be greatly simplified through the hemi-spherical graph updating algorithm of HUGNet [6] (see Algorithm 1). Here, vertices can only receive data via directed edges from the events generated in the past.

However, hemi-spherical EGs have only been demonstrated to be effective for event-based optical flow prediction which is largely based on temporally and geometrically local events. It is not yet clear whether the method will also perform favourably relative to fully-spherical EGs for other problems too - such as classifying graphs. For this study, we use

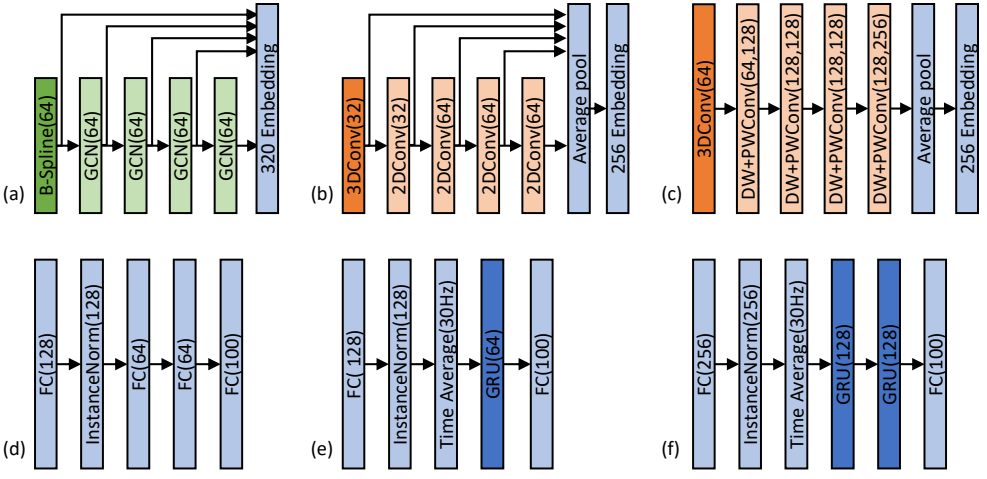

Figure 2: Topologies used in this paper. Top: feature extractors. Bottom: Task heads. ReLU activations not shown for the sake of clarity. Conv-based features extractor also make use of BatchNorm during training. Conv layers use 3x3(x3) kernel size, stride of 1 for the first 2 layers and 2 afterwards. (a) EGNN feature extractor. (b) CNN-Concat feature extractor. (c) CNN-DWPW feature extractor. (d) Feed Forward Task Head (FFTH). (e) Small GRU Task Head (SGTH). (f) Medium GRU Task Head (MGTH).

a time radius r_t of 100 ms, pixel r_{xy} of 10 pixels and form edges only from the 30 closest neighbors. These values were chosen in order to maintain the same aspect ratio of DVS Lip samples (1.2 s in average, 128 pixels) and ensure that most of the vertices are connected to thirty neighbouring vertices.

3.2 Network topologies

3.2.1 EGNN feature extractor

The proposed EGNN feature extractor is inspired from [6] and is depicted in Fig. 2(a). It uses two types of EGNN layers: B-Spline [9] and GCNconv [14]. B-Spline convolution incorporates complex vertex feature sharing mechanism, which leverages spatiotemporal differences between vertices in the edges to modulate message passing. On the other hand, GCNconv convolution is more straightforward whereby features from neighbouring vertices are averaged before being multiplied by a shared weight kernel. Therefore, the exact relative positions of neighbouring vertices do not influence the convolution. Our EGNN feature extractor is composed of an input B-Spline convolutional layer followed by four GCNconv layers. We found this was a good compromise between the powerful, but computationally expensive, B-spline convolution and the more lightweight GCN convolution. We use 7 input features - event coordinates, approximate normal to local event plan and polarity - as in [6]. In each layer, the graph convolutions result in vertices of sixty-four features. The vertex features from each layer are then concatenated together resulting in a 320-size embedding that is fed to the task head.

3.2.2 CNN feature extractor

We design two CNN-based feature extractors in order to compare with the EGNN-based one. For this comparison to be fair, we aim at designing a CNN-based feature extractor

that has the same number of parameters as the EGNN feature extractor. Inspired by [28], both convolutional feature extractors perform a 3D convolution on a volume of events in the first layer which is then followed by series of 2D convolutional layers. Note that this resembles our proposed EGNN feature extractor, whereby a more complex layer (3D conv and B-Spline) is applied to the input data, before light layers (2D conv and GCN) build upon these features. The difference between the two CNN implementations is the way in which feature maps from different layers are handled in order to create the embedding vector fed to the task head as shown in Fig. 2. The first implementation (CNN-Concat) takes inspiration from our EGNN feature extractor, whereby global mean pooling is applied to each of the feature maps in each layer resulting in 256 values which are concatenated together. The second takes a more conventional approach based on depthwise and pointwise convolutions (CNN-DWPW). More details are provided in the supplementary material.

3.2.3 Task heads

The task head is a neural network that takes as input the 320-size vector of the feature extractor and outputs a task dependent prediction. In this case the output is a probability distribution over the 100 possible spoken words. In order to study the ability of EGNNs to capture temporal dependencies we design three task heads. The first one is a Feed Forward Task Head (FFTH), similar to that used in [3]. The embedding of each vertex is processed by a series of fully connected layers and the output probability distribution is computed for all vertices in the graph. These outputs are then averaged over the full input sequence and a prediction is made based on the most probable class. The two other task heads are based on gated recurrent units (GRUs). Here a single embedding is calculated for all events within a given time intervals by performing global average pooling on all vertex features. The size of this time interval is determined by the rate of the GRU and in this paper is typically chosen to be either 30Hz or 75Hz. We study the impact of a Small GRU Task Head (SGTH) and a Medium GRU Task Head (MGTH). The dimensions and number of layers of each head are detailed in Fig.2.

3.3 Supervised classification learning

An EG was built for each of the spoken words in the dataset - each containing all of the events generated as each word is spoken. Each model is trained over 100 epochs using batches of sixteen single word EGs, the AdamW optimisation algorithm [18] and a cross entropy loss. We use a plateau loss scheduling whereby after ten epochs, if the average loss over the epoch has not diminished by 5%, the learning rate is halved. The reported accuracy corresponds to that achieved on the test set after the last epoch of training.

We additionally leverage a variety of data augmentation techniques. The baseline data augmentations applied to the EGs are random temporal distortion, edge dropout [25] and horizontal flip as in [6]. Temporal distortion randomly stretches the EG such that the word is spoken at rate between 50% slower to 50% faster than the original recording. Edge dropout of probability 25% will randomly remove edges in the graph at training time and horizontal flip has a 50% to mirror the input EG along its vertical axis. Dropout is also used before and in the GRU layers with 20% probability. On top of this, we also propose a new data augmentation technique we refer to as Maskout, which we originally devised as a means of mitigating potential overfitting of the GRU layers.

Simply, we divide the input EG into regular temporal intervals of duration T, in sync with GRU frames. We then randomly mask out intervals (i.e., delete the events they contain from

Task Head	Acc test	Parameters	GOPs/s
EGNN + FFTH	53.6 %	132 k	1.35
EGNN + SGTH	55.9 %	156 k	1.10
EGNN + MGTH (**G2N2**)	58.7 %	413 k	1.67

Table 1: Comparison of Task Heads (TH). FF = Feed Forward, SG = Small GRU, MG = Medium GRU. G2N2 represents EGNN + MGTH.

the graph) - although preserving edges between vertices that traverse the resulting empty regions. For each EG we randomly choose N_{mask} regions to maskout where an integer number of sequential regions of duration T is randomly chosen by sampling a number between one and L_{mask}. This permits the masked regions to be of a variable time duration. We also apply all of these data augmentations to the event-frames with the exception of edge dropout.

4 Experiments

4.1 DVS Lip

The DVS-Lip dataset is a machine learning dataset designed for automatic lip-reading using event cameras. The dataset includes 19,871 valid word samples spoken by 40 volunteers (20 male, 20 female). The vocabulary contains a total of 100 words, divided into two parts: visually similar word pairs and common words. In order to evaluate the generalization of trained methods, 30 volunteers are used for training and 10 for evaluation. The dataset includes both the event stream and intensity images (25FPS) output from the event camera, and mouth-centered crops of size 128x128 pixels are extracted using face detection. An example can be seen in Fig. 1.

4.2 Results

We first look at the impact of the task head on the HUGNet EGNN feature extractor - in particular to understand whether the GRU can bring about an improvement in performance due to its ability to better capture temporal dependencies. We then demonstrate the significant performance boost due to our new data augmentation technique Maskout and study the impact of the hemi-spherical update feature extractor relative to the more computationally intensive fully-spherical one.

Finally, using the optimal configuration of our model, we perform a benchmarking against the two CNN based approaches as well as larger CNNs from the state of the art on event-based lip reading.

4.2.1 Task head

Table 1 presents the results on DVS Lip achieved by the HUGNet EGNN feature extractor combined with the three different task heads.

The utility of the GRU in modelling temporal dependencies can be clearly seen in comparing the first two lines of the table. Whereas the FFTH head has access only to the features calculated by the EGNN using the information propagated forward in time between vertices connected by edges, the small GRU task head is able to integrate this information over the full spoken word and obtain a higher test accuracy. The medium sized GRU brings about another increase in performance at the expense of a slight increase in the number of operations per second and the total number of parameters. We observed that further increases in GRU size (i.e., large GRU) only very marginally improved the accuracy beyond that of EGNN + MGTH - which we will refer to as G2N2 in the remainder of the article.

N_{mask}	0	4	4	4	4	6	8	10
L_{mask}	-	4	6	8	10	6	6	6
Acc (%)	58.7	64.1	65.0	65.4	65.2	65.9	**66.1**	63.5

Table 2: Impact of Maskout data augmentation with a hemi-spherical update graph and GRU task head (G2N2) on the accuracy. The GRU frequency is 30 Hz. N_{mask} and L_{mask} are the number of masks and maximum length for the EG respectively and T=33ms.

Neighbor Search	Search radii	Search Volume	Latency	Acc
Fully-spherical	(10 pix, 50 ms)	V	250 ms	68.3 %
HUG	(10 pix, 100 ms)	V	0 ms	**69.4**%

Table 3: Comparison of EG building parameters. Hemi-spherical Update Graph (HUG) [6] vs Spherical Update. GRU is processed at 75 Hz. Latency represents the time between creation and output feature computation for an event.

4.2.2 Event-graph data augmentation

In Table 2 we report the effect of applying our data augmentation scheme Maskout on G2N2 and the impact of its two parameters N_{mask} and L_{mask}. We observed that by applying Maskout, over different combinations of parameters, the test accuracy of G2N2 was dramatically boosted by between 4.8% and 7.6%. The optimal configuration was the application of eight randomly positioned masks with a duration of between 33ms up to 200 ms (six slices of 33ms). The spoken word EGs will often be composed to several isolated EG islands between which message passing is not possible as the maximum length of masks is greater than the search radius, possibly explaining the gains.

4.2.3 Comparison of hemi- and fully-spherical EGs

In order for our EGNN feature extractor to be lightweight, we opted to use the hemi-spherical update graph neural network (HUGNet) [6] as opposed to other, more computationally costly, fully-spherical EG approaches [3, 21, 26]. While HUGNet, despite its reduction in EG complexity, outperformed fully-spherical approaches on optical flow estimation tasks it is important to understand whether this advantage might also generalise to other computer vision tasks and whether it is a detriment to the performance of G2N2. We therefore present in Table 3 a comparison between hemi-spherical (HUG) and fully-spherical (Fully-spherical). In this experiment, we also used a GRU rate of 75Hz since we found this to be advantageous compared to 30Hz, for a relatively small increase in the number of operations per second (see supplementary material for a sweep of the GRU rate).

Intriguingly, as was reported to be the case for optical flow estimation [6], HUGNet outperforms its fully-spherical counterpart by 1.1% indicating a generality of the hemi-spherical EG approach. In order to emphasise advantages of the HUGNet based in terms of delay we also note the latency after an event has been generated before it can be used by the task head. When HUGNet can process in real time, standard approach adds 250 ms since there are five layers.

4.2.4 CNN Benchmarking

We finally compare our optimal G2N2 model to the two convolutional benchmarks of a similar model size as well as other approaches from the state of the art. The results are summarised in Table 4. Neither CNN-Concat nor CNN-DWPW are capable of attaining a performance close to that of G2N2 - with respective gaps of 6.2% and 3.4%. This shows that, for this task, the EGNN is a much more effective feature extractor than CNNs of the same

	Mode	Rate (Hz)	Acc	Parameters	GOPs/s
CNN-Concat (This work)	Event Frame	75	63.2 %	**418 k**	6.57
CNN-DWPW (This work)	Event Frame	75	66.0 %	**405 k**	8.81
G2N2 (This Work)	Event Graph	75 (GRU)	**69.4** %	**413 k**	**1.68**
RN-Net [31]	Event Frame	33	67.5 %	7.5 M	
[20]	Video Frame	25	65.5 %	11.2 M	
ACTION-Net[30]	Event Frame	25	68.8 %	28.1 M	
MSTP[28]	Event Frame	(25,175)	**72.1** %	60.3 M	

Table 4: Comparison of DVS Lip accuracy and network complexity in number of parameters.

size based on event-frames. Furthermore, in comparing the number of operations per second, we see that G2N2 is able to achieve this performance with up to 84% fewer calculations. This can largely be attributed to the sparse and event-driven manner in which EGNNs compute, relative to CNNs which do not exploit the inherent sparsity of event-data.

We include in the same table other solutions to event-based spoken word classification from the state of the art. Remarkably G2N2 outperforms most of these methods despite the drastic reduction in the model size. G2N2 was only bettered by MSTP, by 2.7%, despite the fact it requires over two orders of magnitude more parameters. It should also be noted that, although outperformed by G2N2, CNN-Concat and CNN-DWPW also obtain reasonably competitive performances relative to other larger CNNs. We observed in our experiments that this is in large part due to our data augmentation technique Maskout.

5 Discussion and Perspectives

We have addressed the limitations of EGNNs in capturing long-term dependencies for spoken word classification. Our proposed solution, G2N2, combines EGNNs with GRU task heads, effectively grasping these temporal dependencies and improving performance. Additionally, our data augmentation technique, Maskout, has shown a remarkable performance boost of up to 7.4% for G2N2. Further exploration of augmentation methods and adaptation from other vision domains is warranted for EGs.

Benchmarking EGNNs against CNNs of similar size demonstrated the superior feature extraction capability of EGNNs. This advantage may stem from the preservation of precise spatiotemporal information captured by event cameras, as well as the distinct characteristics of kernel sizes in CNNs and search radii in EGNNs. Furthermore, EGNNs significantly reduced the number of calculation operations per second compared to CNNs, benefiting from the sparsity of event-based data and avoiding redundant measurements.

G2N2's lightweight nature and the use of a hemi-spherical update EGNN contribute to its computational efficiency compared to fully-spherical EG feature extractors. We observed that G2N2 with a hemi-spherical feature extractor consistently outperformed its fully-spherical counterpart, highlighting the need for further investigation into this phenomenon.

However, the utilization of GRU layers in G2N2 requires a return to the frame-based paradigm for the task head, introducing latency overhead and limiting fully asynchronous inference. Future work will focus on exploring alternatives to GRU task heads to achieve a lightweight and ultra-low latency implementation without relying on frame buffering.

Acknowledgements This work is partly funded thanks to the French national program "Programme d'Investissements d'Avenir, IRT Nanoelec" ANR-10-AIRT-05. This publication was made possible by the use of the FactoryIA supercomputer, financially supported by the Ile-De-France Regional Council.

References

[1] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, Jeff Kusnitz, Michael Debole, Steve Esser, Tobi Delbruck, Myron Flickner, and Dharmendra Modha. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[2] Raymond Baldwin, Ruixu Liu, Mohammed Mutlaq Almatrafi, Vijayan K Asari, and Keigo Hirakawa. Time-ordered recent event (tore) volumes for event cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[3] Yin Bi, Aaron Chadha, Alhabib Abbas, Eirina Bourtsoulatze, and Yiannis Andreopoulos. Graph-based object classification for neuromorphic vision sensing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 491–501, 2019.

[4] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. A differentiable recurrent surface for asynchronous event-based data. In *European Conference on Computer Vision*, pages 136–152. Springer, 2020.

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[6] 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.

[7] Matthew Dutson and Mohit Gupta. Event neural networks. *CoRR*, abs/2112.00891, 2021. URL https://arxiv.org/abs/2112.00891.

[8] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *Proceedings of the International Conference on Learning Representation*, 2019.

[9] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 869–877, 2018.

[10] Thomas Finateu, Atsumi Niwa, Daniel Matolin, Koya Tsuchimoto, Andrea Mascheroni, Etienne Reynaud, Pooria Mostafalu, Frederick Brady, Ludovic Chotard, Florian LeGoff, Hirotsugu Takahashi, Hayato Wakabayashi, Yusuke Oike, and Christoph Posch. A 1280×720 back-illuminated stacked temporal contrast event-based vision sensor with 4.86µm pixels, 1.066geps readout, programmable event-rate controller and compressive data-formatting pipeline. In *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, pages 112–114, 2020. doi: 10.1109/ISSCC19947.2020.9063149.

[11] Daniel Gehrig, Antonio Loquercio, Konstantinos G Derpanis, and Davide Scaramuzza. End-to-end learning of representations for asynchronous event-based data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5633–5643, 2019.

[12] Mathias Gehrig, Mario Millhäusler, Daniel Gehrig, and Davide Scaramuzza. E-raft: Dense optical flow from event cameras. In *International Conference on 3D Vision (3DV)*, 2021.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[14] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *Proceedings of the International Conference on Learning Representation*, 2016.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[16] Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E Shi, and Ryad B Benosman. Hots: a hierarchy of event-based time-surfaces for pattern recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1346–1359, 2016.

[17] Min Liu and Tobi Delbruck. Adaptive time-slice block-matching optical flow algorithm for dynamic vision sensors. 2018.

[18] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017. URL http://arxiv.org/abs/1711.05101.

[19] Ana I Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso García, and Davide Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5419–5427, 2018.

[20] Brais Martínez, Pingchuan Ma, Stavros Petridis, and Maja Pantic. Lipreading using temporal convolutional networks. *CoRR*, abs/2001.08702, 2020. URL https://arxiv.org/abs/2001.08702.

[21] Anton Mitrokhin, Zhiyuan Hua, Cornelia Fermuller, and Yiannis Aloimonos. Learning visual motion segmentation using event surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14414–14423, 2020.

[22] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[23] Atsumi Niwa, Futa Mochizuki, Raphael Berner, Takuya Maruyarma, Toshio Terano, Kenichi Takamiya, Yasutaka Kimura, Kyoji Mizoguchi, Takahiro Miyazaki, Shun Kaizu, et al. A 2.97 μm-pitch event-based vision sensor with shared pixel front-end circuitry and low-noise intensity readout mode. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 4–6. IEEE, 2023.

[24] Etienne Perot, Pierre De Tournemire, Davide Nitti, Jonathan Masci, and Amos Sironi. Learning to detect objects with a 1 megapixel event camera. *Advances in Neural Information Processing Systems*, 33:16639–16652, 2020.

[25] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *Proceedings of the International Conference on Learning Representation*, 2019.

[26] 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.

[27] Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. Hats: Histograms of averaged time surfaces for robust event-based object classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1731–1740, 2018.

[28] 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.

[29] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *Proceedings of the International Conference on Learning Representations*, 2017.

[30] Zhengwei Wang, Qi She, and Aljosa Smolic. Action-net: Multipath excitation for action recognition. *CoRR*, abs/2103.07372, 2021. URL https://arxiv.org/abs/2103.07372.

[31] Sangmin Yoo, Eric Yeu-Jer Lee, Ziyu Wang, Xinxin Wang, and Wei D. Lu. Rn-net: Reservoir nodes-enabled neuromorphic vision sensing network, 2023.

[32] Friedemann Zenke and Surya Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541, 2018.

[33] Kun Zhou, Qiming Hou, Rui Wang, and Baining Guo. Real-time kd-tree construction on graphics hardware. *ACM Transactions on Graphics (TOG)*, 27(5):1–11, 2008.

[34] Alex Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Ev-flownet: Self-supervised optical flow estimation for event-based cameras. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.062.