Efficient Learning of Event-based Dense Representation using Hierarchical Memories with Adaptive Update

Uday Kamal and Saibal Mukhopadhyay

Georgia Institute of Technology, Atlanta, GA, USA
{ukamal6,smukhopadhyay6}@gatech.edu

Abstract. Leveraging the high temporal resolution of an event-based camera requires highly efficient event-by-event processing. However, dense prediction tasks require explicit pixel-level association, which is challenging for event-based processing frameworks. Existing works aggregate the events into a static frame-like representation at the cost of a much slower processing rate and high compute cost. To address this challenge, this work introduces an event-based spatiotemporal representation learning framework for efficiently solving dense prediction tasks. We uniquely handle the sparse, asynchronous events using an unstructured, set-based approach and project them into a hierarchically organized multi-level latent memory space that preserves the pixel-level structure. Low-level event streams are dynamically encoded into these latent structures through an explicit attention-based spatial association. Unlike existing works that update these memory stacks at a fixed rate, we introduce a data-adaptive update rate that recurrently keeps track of the past memory states and learns to update the corresponding memory stacks only when it has substantial new information, thereby improving the overall compute latency. Our method consistently achieves competitive performance across different event-based dense prediction tasks while ensuring much lower latency compared to the existing methods.

Keywords: Event-Based Vision, Memory-Augmented Representation Learning, Efficient Processing, Event-based dense prediction

1 Introduction

Event-based cameras [5, 10, 35, 35], which are bioinspired and aim to mimic the brains efficient data processing, represent a departure from the traditional frame-based cameras. They feature pixels that independently record asynchronous events, capturing brightness changes with high temporal precision (microsecond order) and offering substantial dynamic range (>120 dB). Their low power usage and swift response are particularly beneficial in scenarios with rapid motion or sudden light shifts, such as drones or autonomous vehicles [9, 15, 16, 36, 42]. Nonetheless, these advantages necessitate the development of specialized processing architectures to fully harness their capabilities.

Existing learning-based approaches for event data have primarily relied on grouping events into synchronized stacks before feeding them into convolutional neural networks (CNNs) [4, 7, 13]. However, this strategy not only undermines the high temporal resolution and asynchronicity inherent in event data, but also offers a limited temporal context to the foregoing recurrent neural networks (RNNs) [14, 19]. Recent endeavors in event-based processing techniques for object recognition—such as time-surface representations [23], 3D event-clouds [39], and graph-based methods [25]—have mostly prioritized sparse processing for computational efficiency. Yet, they fall short in performance when compared to their dense-representation counterparts [32].

Efficient processing of event-triggered asynchronous data necessitates algorithms that construct and continuously refine an internal representation, correlating new and historical events across both space and time. Although recurrent modules per pixel could trace past events [7], this approach significantly increases processing and memory requirements with higher spatial resolution. Recently proposed work [21] has shown impressive compute efficiency due to its associative memory augmented recurrent representation learning that preserves sparsity and enables event-by-event spatiotemporal correlation. However, their method has only been evaluated on simple classification tasks and it is non-trivial to scale it on dense tasks due to the lack of explicit pixel-level association to the memory states and its reliance on a single-stage memory. A hierarchical latent memory-based method has been proposed [17] that encodes the event representation into a stack of memory states with a multi-rate update mechanism. Despite achieving superior performance across diverse dense tasks, this method still relies on a hand-tuned update rate for the memory stacks limiting its adaptability and optimality in real-time applications. Moreover, unlike EventFormer [21], they do not consider the inherent recurrent nature of their memory states that can further avoid redundant updates, and improve latency. While it is desired to change this update rate adaptively depending on the input scenario, it is not trivial to learn them directly since it is still required to keep track of the states in the absence of an update. At the same time, computing the association between two subsequent memory levels for each time step can be compute-intensive, especially for earlier memory levels with higher spatial resolution. Essentially, we need a recurrent mechanism that efficiently builds up representation with incoming events and triggers an update only when it has accumulated sufficient new information.

To this end, we present a novel architecture for efficient processing of event-based dense tasks. Our architecture introduces recurrent processing of memory-augmented representation learning by leveraging a multi-stage memory structure that enables gradual feature enhancement. Here, the lowest memory level encodes the local dynamic information by projecting the unstructured, sparse events into a spatially structured, dense latent space. Subsequent memory levels learn to extract higher-level static and global contexts. Instead of updating the whole compute stack at each time step, our method uses an adaptive memory update mechanism that can learn the optimal update rates from data, thereby

enhancing the models responsiveness to the dynamic scenes and its overall efficiency. Moreover, rather than using the entire set of memory states to perform an association between two subsequent levels, we introduce a global pooling operation to compress the memory states, recurrently process them and adaptively skip updates in the absence of any substantial new information. Effectively, this operation saves computation and avoids redundant updates due to any low-level information gain. In other words, we avoid updating if the changes in the memory states between events are minor in the global context. To the best of our knowledge, we are the first to introduce an adaptive component to event-based memory models, which crucially improves the computational load required for its real-world application.

We evaluate our method on several event-based prediction tasks and datasets that have traditionally leveraged dense information, including semantic segmentation, object detection, and depth estimation. Our method demonstrates superior reduction in latency ($50\% - 70\%$) while maintaining competitive task performance, compared to state-of-the-art baselines.

To summarize, our contributions are:

- We introduce an end-to-end learning framework with an adaptive update rate for multi-stage memory architecture.
- We propose a global context-based association between two contiguous memory levels to efficiently process their recurrent updates.
- We experimentally show that our method achieves a significant reduction in latency while maintaining competitive performance on different event-based dense perception tasks.

2 Related Works

We discuss the relevant prior works from two different directions: building an effective representation for event camera data and extracting useful features from this representation for the downstream tasks.

Event representation: Building an effective representation for event-based data is one of the key challenges for its efficient processing mostly due to its highly sparse and asynchronous nature. Both manually designed and data-adaptive representation space have been explored in several prior works that consider aggregating event information into a dense, frame-like structure. Histogram-based methods, such as hierarchy of event-based time surface (HOTS) [23], and histograms of averaged time surface (HATS) [34], encode the timestamp information of the events by computing a time-ordered representation. However, their handcrafted nature of event encoding can result in sub-optimal representation. To this end, data-adaptive methods have been proposed that rely on end-to-end learnable representation. Event spike tensor (EST) [13] introduced learnable kernels for histogram-based methods, further improving performance. Voxel Grid-based representation has also been considered where AED [26] and ASTMNet [24] introduced data-adaptive kernel and sampling mechanism. However, they need to

keep track of past events for temporal correlation and perform redundant computations every time a new event occurs. Memory-augmented representation can overcome this challenge by internally tracking past events in its memory states. Matrix-LSTM [7] proposed a pixel-wise, LSTM-based memory model that keeps track of past events at each pixel location independently. However, such representation does not consider any spatial correlation and, therefore requires a compute-expensive feature extractor. EventFormer [21] addresses this challenge by proposing a set-based spatiotemporal association while building the memory states. However, scaling EventFormer for dense prediction is challenging due to the absence of any explicit correspondence between the memory states and the pixel location.

Event feature extraction: Extracting useful features from the event representation is crucial for the downstream task. Popular feature extraction architectures like CNNs, which are originally designed for frame-based data, have been extensively explored in prior works [4, 28]. These methods process discretized events aggregated into a dense representation, discarding their inherent sparsity and temporal correlation. Several new directions have been explored to leverage the sparse and asynchronous nature of the event data, including sparse asynchronous CNN architecture [29] and graph neural networks [25, 32]. Despite their superior compute efficiency, graph-based methods have shown limited performance compared to the dense, synchronous methods on complex tasks [32]. MLP and Transformer-based architectures have also been explored in this direction. These methods consider events as streams of space-time point clouds and leverage simple MLP-based PointNet [33, 38] or Transformer [31] architectures to extract useful features. Although these methods have shown superior performance on simple classification tasks, it is challenging to scale them up for complex, dense tasks while maintaining event-by-event processing. To address this challenge, a hierarchical latent memory-based solution has been proposed [17] that encodes raw events into a structured memory space and leverages it for downstream dense tasks. Despite performing well on several dense tasks, this method relies on a hand-tuned, fixed update rate without considering the relative information gain with respect to its past update, which perform suboptimally and cause redundant computation.

3 Method

3.1 Preliminaries:

An event-based camera contains independent sensors at every pixel location that can asynchronously respond to the brightness change [11]. Specifically, within a time interval τ, events produced at pixel locations $(x, y)_k$ can be mathematically expressed as, $\mathcal{E}_\tau = \{(x_k, y_k, t_k, p_k) \mid t_k \in \tau, k \in \mathcal{N}\}$ where t_k is the event timestamps, $p_k \in \{-1, +1\}$ corresponds to relative change in brightness, and $\mathcal{N}$ denotes the total number of events. We want to learn a parametric mapping $\mathcal{F} : \mathcal{E}_\tau \rightarrow \{\mathcal{M}^i_{\mathcal{E}_\tau}\}$ to encode the unstructured list of events into a stack of

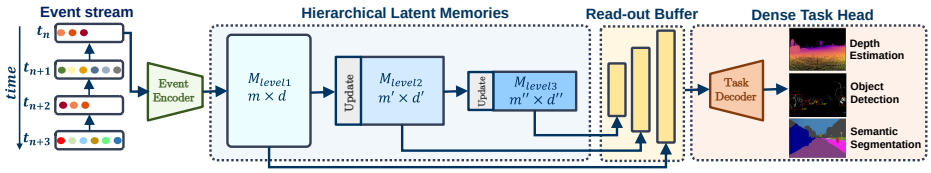


Fig. 1: Overview of the overall architecture. It consists of an event encoder that directly maps the raw events into the lowest level of latent memory (M_{level1}) at each time step. Subsequent memory levels are updated adaptively by utilizing the previous memory states. The readout buffer extracts and stores the memory states to be used by the task-specific decoder at the end.

hierarchically structured memory representation $\{\mathcal{M}^i_{\mathcal{E}_\tau}\}$ where i denotes the level of hierarchy that can be directly integrated with off-the-shelf task heads (decoders) to solve event-based dense tasks. We also want $\mathcal{F}$ to be capable of updating its memory representation adaptively (i.e. only when there is sufficient new information) to avoid redundant computation.

3.2 Overview of the framework

We leverage a hierarchy of latent memory stacks to extract high-level contextual information from the low-level event streams. Specifically, spatially unstructured event streams are first projected into the lowest level of structured latent memory space, which is followed by multiple higher memory levels. Fig. 1 shows a high-level overview of the method. Since very low-level events most often do not contribute much to the higher level of representation, we propose a learnable memory update mechanism that allows higher levels of memories to select between preserving or updating its past states, thereby resulting in an adaptive-rate network architecture and saving computation. We introduce a compressed-space memory association to compare the global context of subsequent memory states and leverage a recurrent module to keep track of the associations. The updated memory states are extracted and stored in its corresponding readout buffer. A task-specific decoder at the final layers exploits these stored buffer states at each time step and computes prediction.

Event representation and encoding: We consider window-based pixel association of the events [17]. Specifically, the lowest level memory $M_{level1} \in \mathbb{R}^{m \times D}$ contains m rows where each row stores $D-$dimensional representation from a $w \times w$ sized window (with a stride of w) at a spatial location (i, j). At any time step t_n, with N number of events triggered within the time-interval of $[t_{n-1}, t_n]$, we compute the relative position $(\hat{x_k}, \hat{y_k})$ within the window w_{ij}, and relative timestamp $\hat{t}_k$ (within the interval of $[t_{n-1}, t_n]$) for each event. This low-level event information is then projected into a $D-$dimensional embedding vector π_k using the following transformations:

$$\pi_k = LN([F_{pos}(\hat{x}_k, \hat{y}_k),\ \ F_t(\hat{t}_k),\ \ F_p(p_k)]), \tag{1}$$

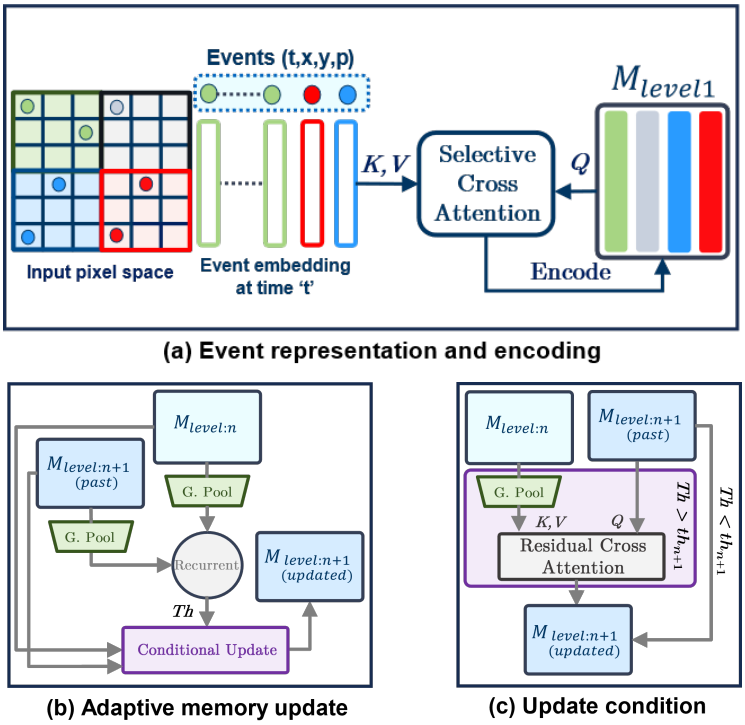


Fig. 2: Details of the different components of the architecture. (a) Embeddings computed from raw events are spatially encoded into the lowest memory level using cross-attention, (b) Global states from two subsequent memory levels are sequentially processed to generate score (Th) for the adaptive update, (c) memory states are either updated using residual cross attention or retained (therefore bypassing expensive operation) depending on the Th value.

where F_{pos}, F_t, F_p each denotes an MLP with two hidden layers for event position, timestamp, and polarity, respectively; LN denotes Layer Normalization [3]; and $[,]$ denotes concatenation operation. This transformed event representation is next encoded to the memory cell using cross-attention, as shown in Fig. **??**(a). Specifically, we compute cross attention between a memory cell s_{ij} in M_{level1} and all the n-event embeddings $\pi \in \mathbb{R}^{n \times D}$ that fall within the spatial window w_{ij}, therefore selectively updating only those memory cells that receive any events:

$$\begin{aligned} q_{ij} = \mathcal{Q}(s_{ij}), \quad K = \mathcal{K}(\pi), \quad V = \mathcal{V}(\pi) \\ s_{ij}^{new} = \text{softmax}\left(\frac{q_{ij}K^T}{\sqrt{D}}V\right) \end{aligned} \tag{2}$$

where $\mathcal{Q}, \mathcal{K}, \mathcal{V}$ are the MLP-based mapping for computing query, key, and value, respectively.

Adaptive update of the memory: The hierarchical memory structure requires the update to the next memory level to be conditioned on the previous memory states. We achieve this by computing a window-based multi-head cross-attention (WCA) [17, 27] between two subsequent memory levels. We compute the *Query* from the current memory level (M_{n+1}) while the *Key* and *Value* from a downsampled version of the previous memory level (M_n):

$$\begin{aligned} \hat{M}_{n+1} &= WCA(M_{n+1},\ \text{Down}(M_n) + M_{n+1} \\ M_{n+1}^{updated} &= \text{MLP}(\text{LN}(\hat{M}_{n+1})) + \hat{M}_{n+1} \end{aligned} \tag{3}$$

where MLP denotes two conv 1×1 layers with GELU activation [18], and Down is a spatially downsampling operation with strided convolution. We avoid redundant updates to the memories by introducing a learnable update score, Th. Specifically, we consider the global representation of two subsequent memory levels and recurrently compute the update score for each time step:

$$\begin{aligned} M_{n+1}^G = \text{G.Pool}(M_{n+1}), \quad M_n^G = \text{G.Pool}(M_n) \\ \hat{Th} = \mathcal{R}(M_{n+1}^G, M_n^G), \quad Th = \text{sigmoid}(\text{MLP}(\hat{Th})) \end{aligned} \tag{4}$$

where G.Pool denotes the global pooling operation G.Pool : $M_{\{.\}} \in \mathbb{R}^{m \times d} \rightarrow M_{\{.\}}^G \in \mathbb{R}^{1 \times d}$, $\mathcal{R}$ denotes the recurrent operation (a GRU unit), and MLP here denotes a linear mapping to a scalar value. Subsequent memory update occurs only when this Th exceeds a certain threshold (th). We also reset the hidden state of $\mathcal{R}$ following each update.

Readout buffers: Readout buffers [17] extract the memory states and utilize a point-wise convolution layer to further refine them to be utilized by the task head at the end at every time step. The operation is defined as:

$$ro_n = c_{ro}(LN(M_n))) \tag{5}$$

where ro_n, and M_n are the refined buffer state, and memory states at level n, respectively; c_{ro} is a conv 1×1 layer with Group Normalization [40] and SiLU activation [8].

4 Experiments

Base training and inference setup: We consider three different event-based dense prediction tasks for experiments: segmentation, object detection, and depth estimation. To evaluate our performance, we use both the algorithmic performance (task-specific metrics) and the inference latency. For our network configuration, we use three memory levels with $D_i \in \{32, 64, 128\}$ (with increasing memory dimension for higher levels) and window stride size $s_i \in \{4, 8, 16\}$. Unless otherwise specified, we use 4 as the number of heads for all cross-attention operations, and the 0.5 as the threshold (th) for adaptive update. For training, we use Adam [22] as our optimizer with a batch size of 4 and an initial learning

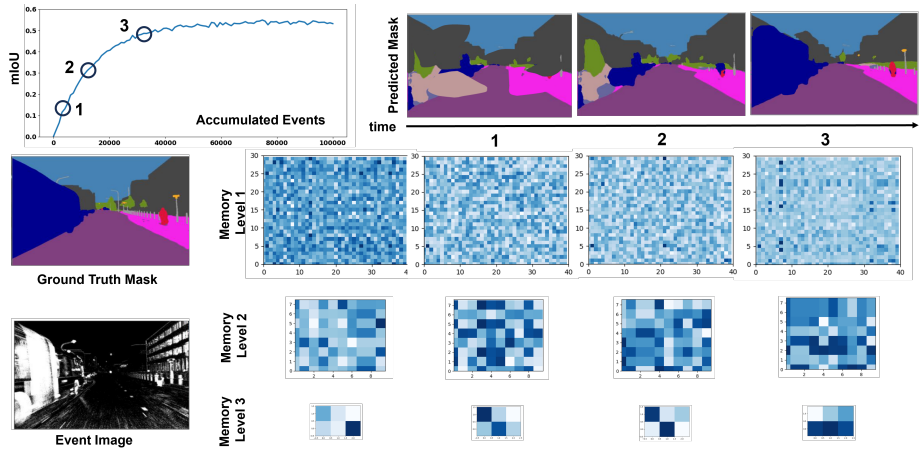


Fig. 3: Temporal evolution of the memory states across different levels. The prediction performance gradually improves as we get more and more events while the memory states tend to form distinct patterns.

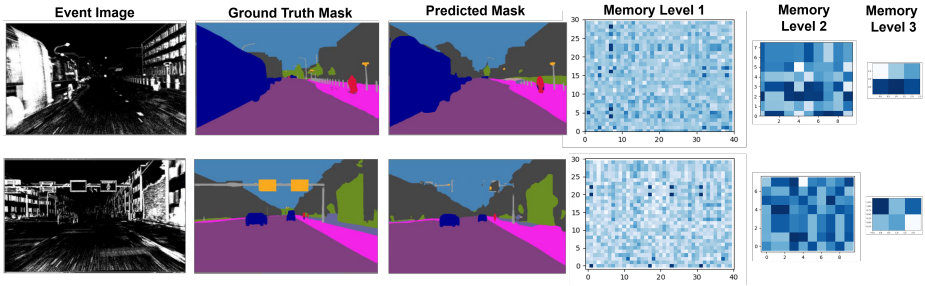


Fig. 4: Visualization of memory states for different inputs where we observe unique patterns but similar trends (sparse to dense) across different memory levels.

rate (LR) of 5.0e-4 with a cosine LR scheduler. For fair comparison with other methods, we use the same hardware setup as [17] with NVIDIA Tesla V100 GPU to report latency.

4.1 Qualitative analysis

Temporal evolution of multi-scale memory: We first visualize the temporal evolution of the latent memory states of our proposed method trained on DSEC-Semantic dataset [15] as shown in Fig. 3) where we plot the first channel of each memory level. We observe that all memory states start from an initial state but as we get more and more events, the states get more refined and start to form a distinct pattern. Specifically, the lower level memory tends to form a sparse

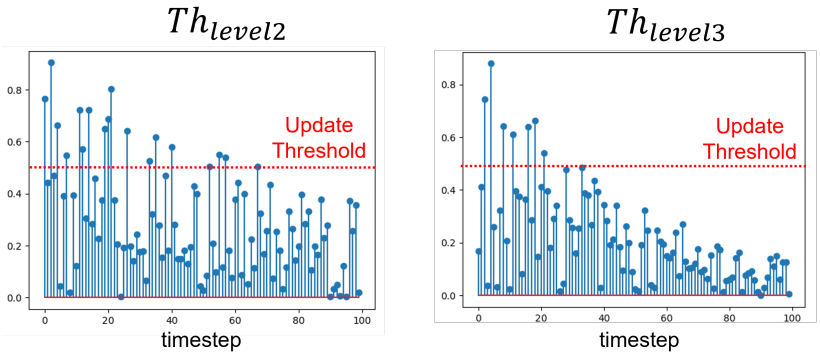


Fig. 5: Learned memory update score across different levels where the score gradually decays over time with a higher decay rate for higher memory levels.

representation while the higher levels tend to form a more dense representation gradually. This can be explained by the fact that the higher resolution of the lower-level memory captures more coarse-grained features which get more refined throughout the stacks. This is evident from the gradual improvement in both quantitative (mIoU) and qualitative results (predicted mask) as we get more and more events. We also visualize the converged memory pattern for two different input samples in Fig. 4 where we observe visibly distinguishable patterns in the state representation with a similar sparse to dense pattern formation trend across the memory stacks.

Adaptive multi-scale update across hierarchies: We analyze the temporal adaptation of the learned update scores, Th_{level_n}, across the memory stacks of our proposed method trained on the DSEC-Semantic dataset as shown in Fig. 5). We observe that an update to the higher level memory is not necessary throughout the whole observation period where most of the updates occur during the initial period of observation. Our method adaptively learns to have fewer updates to the higher-level memory compared to the lower ones. The intuition behind this is that for the static dense prediction task, the higher level memory captures more fine-grained, global information that requires less frequent updates as we get more and more events. The proposed recurrent formulation ensures keeping track of the skipped updates so that they can influence future ones. We also show in Section 4.3 that the proposed adaptive update mechanism ensures faster convergence to the memory states for higher-level memory, thereby reducing redundant computation and improving the overall latency.

4.2 Quantitative results

The section describes the quantitative experimental results of our method compared to the existing methods on several event-based dense prediction tasks including semantic segmentation, object detection and depth estimation.

Table 1: Results of (a) semantic segmentation task on DSEC-Semantic dataset, and (b) object detection on GEN1 dataset. Latency values are measured on the TeslaV100 GPU system.

Method	Decoder	mIoU	Latency (ms)
EV-SegNet [1]	UNet	51.8	-
ESS [37]	UNet	53.3	16.2
HMNet-B3 [17]	UPerNet	53.9	9.7
HMNet-L3 [17]	UPerNet	57.1	13.9
Ours	UPerNet	49.9	4.5

(a)

Method	Decoder	mAP	Latency (ms)
MatrixLSTM [7]	YOLOv3	31.0	-
NGA [20]	YOLOv3	35.9	-
RED [30]	SSD	40.0	11.6
Asynet [6]	YOLO	12.9	-
AEGNN [32]	YOLO	16.3	-
AED [26]	YOLOX	45.4	13.1
ASTMNet [24]	SSD	46.7	35.6
HMNet-B3 [17]	YOLOX	45.2	7.0
HMNet-L3 [17]	YOLOX	47.1	7.9
Ours	YOLOX	44.8	3.2

(b)

Semantic segmentation: We conduct event-based semantic segmentation experiments on the DSEC-Semantic dataset [15] with 640×480 pixel resolution event-camera data recorded under real-world street scenes. The pixel-wise semantic segmentation annotation for this dataset are recorded from RGB images at 20 Hz with a total 8,082 and 2,809 frames available for training and testing, respectively. We follow the same setup in [17] with 11 object classes for our experiments and UPerNet [41] as the task head. We train this model for 50k iterations with our base training setup. Table 1b shows the result. ESS [37] adopts additional finetuning on RGB-based data while our method relies on event data only. HMNetB3/L3 achieves the best-performing mIoU but they incur higher latency mostly due to the redundant memory updates at a higher level. Our method achieves a balance between computation and task-specific performance with a competitive mIoU score while outperforming all the existing methods in terms of latency, with a 50%/70% improvement over the SoTA methods HMNetB3/L3.

Object detection: We conduct event-based object detection experiments on GEN1 dataset [30] with 304×240 pixel resolution event-camera data recorded under a real-world driving scenario. This dataset includes 2,358 event sequences each with 60 sec time duration. These sequences are further divided into 1,459, 429, and 470 for training, validation, and testing, respectively. The bounding box annotations are available at 1Hz to 4Hz, depending on the sequence. There are two object level annotations: pedestrian and car. We adopt the same YOLOX [12] detector head as used in [17], and train the model end to end for 300k iterations. Table 1a shows the comparative results with the existing methods. Our method outperforms most of the recurrent baselines (MatrixLSTM, and RED) both in terms of mAP and latency. The best-performing recurrent method ASTMNet requires significant compute latency mostly due to its long accumulation time. Our

Table 2: Results of depth estimation task on MVSEC dataset. Latency values are measured on the TeslaV100 GPU system.

Method	Decoder	outdoor day1			outdoor nigh1			Latency (ms)
		REL ↓	RMS↓	RMSlog↓	REL↓	RMS ↓	RMSlog↓	
E2Depth [19]	UNet	0.346	8.564	0.421	0.591	11.210	0.646	-
RAMNet [14]	UNet	0.303	8.526	0.424	0.583	13.340	0.830	9.0
HMNet-B3 [17]	UNet	0.270	7.101	0.332	0.323	8.935	0.462	5.0
HMNet-L3 [17]	UNet	0.254	6.890	0.319	0.323	9.008	0.482	6.9
Ours	UNet	0.292	7.985	0.386	0.358	9.441	0.498	2.3

method achieves competitive performance against these methods while reducing 60% latency of the existing SoTA method (HMNet-L3).

Depth estimation: We conduct event-based depth estimation experiments on real MVSEC dataset [42] with the street-scene recording of 346×260 pixel resolution DAVIS event-camera data. A LiDAR sensor is used to record the ground truth depth map at 20 Hz. We follow the same training and testing pipeline in [17] with the outdoor day2 sequences for training and outdoor day1 and outdoor night1 sequences for evaluation. We adopt the pre-training pipeline proposed in [14] and the UNet-based decoder in [17]. Table 2 shows the comparative results with the existing methods. Although RAMNet leverages a fusion of event and image space, our method can outperform it on both day and night sequences while achieving 74% latency improvement. While HMNet-B3/L3 with fixed update achieves the best result, our method with dynamic update can still perform competitively while reducing the latency by 50%/67%.

4.3 Ablation experiments

One of our key contributions is the adaptive update to the memory states. A straightforward alternative is having a fixed, uniform update rate across all the memory levels. To better understand the effect of adaptation, we compare the performance of the model with adaptive updates to that of the model without it (i.e. replacing it with the uniform update mechanism). We conduct several ablation experiments for one of the vision tasks – specifically, segmentation with the DSEC-semantic dataset. Our method also relies on a threshold parameter (th) to determine the update decision, which can make our method sensitive to this parameter. Therefore, we also conduct a sensitivity analysis of this parameter.

Adaptive vs. uniform memory update: We investigate the impact of the proposed adaptive memory update mechanism in Fig. 6. We compute the mean L_1 norm of the difference between two successive memory states at each level and normalize them to quantitatively analyze the change in memory states that

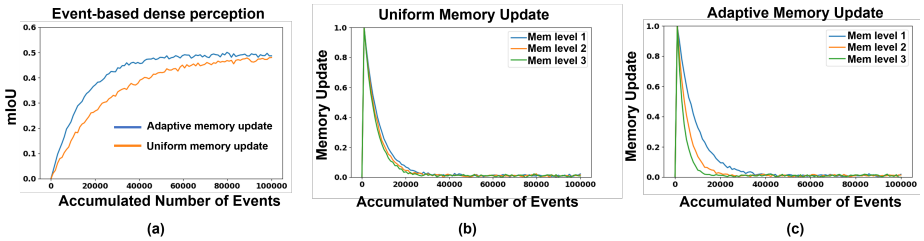


Fig. 6: Comparison between adaptive and uniform memory update on DSEC-Semantic dataset: (a) event-based dense prediction results, and visualization of memory update across different levels for (b) uniform and (c) adaptive memory update, respectively. Here *Memory Update* denotes the mean L_1 norm between two successive memory states normalized for each level.

Table 3: Ablation results on (a) adaptive vs uniform update (b) sensitivity analysis on the update threshold (th) on DSEC-Semantic dataset

Method	mIoU	Latency (ms)
Ours (Uniform update)	49.7	5.8
Ours (Adaptive update)	49.9	4.5

(a)

Method	Update threshold	mIoU	Latency (ms)
Ours (Adaptive update)	0.5	49.9	4.5
	0.3	49.8	4.8
	0.7	46.8	4.1

(b)

occurred with the incoming events. As a baseline, we compare it against a uniform update rate across all hierarchies. We observe that adaptive memory update enables higher levels of memory to converge at a much faster rate, thereby reducing redundant computation (Fig. 6 (c)). On the other hand, there is not much difference in convergence rates for the uniform update(Fig. 6 (b)). Updating higher levels of memory at the same rate as the lowest one can introduce not only redundant computation but also noisy updates, which can slow down the convergence rate and degrade performance, as we can observe in 6 (a)) and Table 3a. The Hierarchical memory model with a uniform update slightly degrades the performance while increasing the overall latency by 28%.

Sensitivity analysis of update threshold: We conduct a sensitivity analysis of the update threshold for our proposed adaptive memory update score Th_{level_n} where we vary the threshold by ± 0.2. Table 3b shows the result. We observe that the overall latency has a profound impact on the threshold since it directly controls the number of updates to the memory (lower update count and improved latency with a higher threshold value). However, increasing the threshold too much can degrade performance as shown in the result. A higher threshold value

may prevent updates necessary for learning an effective representation, therefore harming the performance.

5 Future Directions

One of the limitations of our proposed method is its sensitivity to the threshold parameter for memory update as shown in Section 4.3. Determining the optimal threshold can be non-trivial, especially if higher memory levels were to be used in the model. One potential solution to this could be learning this threshold in an end-to-end manner [2].

Our proposed adaptive update mechanism currently relies primarily on a single modality (event-camera data). Multimodal fusion between events and frames has been explored in recent works [14, 17]. However, it is non-trivial to directly apply our method in a multimodal setting since different modality needs to incorporate different update rates. An interesting future direction could be learning modality-conditioned adaptive updates while fusing different modes together in a compute-efficient manner.

6 Conclusion

In this work, we introduce a novel adaptive update-based hierarchical memory network for low latency event-based dense perception. The proposed method processes the sparse, unstructured raw events and directly encodes them into a compact latent memory space while preserving sparsity. To avoid redundant memory operations and improve latency, we propose a recurrent update mechanism that learns to adaptively update the memory states only when there is sufficient new information. Results on different event-based dense prediction tasks demonstrate the effectiveness of the proposed method. Future works can explore novel adaptive update mechanisms to enable efficient multimodal fusion such as fusion between events and frames.

7 Acknowledgements

This work is supported by the Defense Advanced Research Projects Agency (DARPA) under Grant Numbers GR00019153. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Department of Defence, DARPA, or the U.S. Government.

The corresponding author would like to thank Saurabh Das for his feedback during the early stage of this project and acknowledge Eley Ng for her insightful discussion while preparing this manuscript.

References

1. Alonso, I., Murillo, A.C.: Ev-segnet: Semantic segmentation for event-based cameras. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2019) 10
2. Azarian, K., Bhalgat, Y., Lee, J., Blankevoort, T.: Learned threshold pruning. arXiv preprint arXiv:2003.00075 (2020) 13
3. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016) 6
4. Baldwin, R., Liu, R., Almatrafi, M.M., Asari, V.K., Hirakawa, K.: Time-ordered recent event (tore) volumes for event cameras. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022) 2, 4
5. Brandli, C., Berner, R., Yang, M., Liu, S.C., Delbruck, T.: A 240 × 180 130 db 3 µs latency global shutter spatiotemporal vision sensor. IEEE Journal of Solid-State Circuits **49**(10), 2333–2341 (2014). https://doi.org/10.1109/JSSC.2014.2342715 1
6. Cannici, M., Ciccone, M., Romanoni, A., Matteucci, M.: Asynchronous convolutional networks for object detection in neuromorphic cameras. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2019) 10
7. Cannici, M., Ciccone, M., Romanoni, A., Matteucci, M.: Matrix-lstm: a differentiable recurrent surface for asynchronous event-based data. In: Proceedings of the European Conference on Computer Vision (ECCV), Glasgow, UK. pp. 23–28 (2020) 2, 4, 10
8. Elfwing, S., Uchibe, E., Doya, K.: Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. Neural networks **107**, 3–11 (2018) 7
9. Falanga, D., Kleber, K., Scaramuzza, D.: Dynamic obstacle avoidance for quadrotors with event cameras. Science Robotics **5**(40), eaaz9712 (2020) 1
10. Finateu, T., Niwa, A., Matolin, D., Tsuchimoto, K., Mascheroni, A., Reynaud, E., Mostafalu, P., Brady, F., Chotard, L., LeGoff, F., Takahashi, H., Wakabayashi, H., Oike, Y., Posch, C.: 5.10 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). pp. 112–114 (2020). https://doi.org/10.1109/ISSCC19947.2020.9063149 1
11. Gallego, G., Delbrück, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A.J., Conradt, J., Daniilidis, K., et al.: Event-based vision: A survey. IEEE transactions on pattern analysis and machine intelligence **44**(1), 154–180 (2020) 4
12. Ge, Z., Liu, S., Wang, F., Li, Z., Sun, J.: Yolox: Exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430 (2021) 10
13. Gehrig, D., Loquercio, A., Derpanis, K.G., Scaramuzza, D.: End-to-end learning of representations for asynchronous event-based data. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5633–5643 (2019) 2, 3
14. Gehrig, D., Rüegg, M., Gehrig, M., Hidalgo-Carrió, J., Scaramuzza, D.: Combining events and frames using recurrent asynchronous multimodal networks for monocular depth prediction. IEEE Robotics and Automation Letters **6**(2), 2822–2829 (2021) 2, 11, 13

15. Gehrig, M., Aarents, W., Gehrig, D., Scaramuzza, D.: Dsec: A stereo event camera dataset for driving scenarios. IEEE Robotics and Automation Letters **6**(3), 4947–4954 (2021) 1, 8, 10
16. Hagenaars, J.J., Paredes-Vallés, F., Bohté, S.M., De Croon, G.C.: Evolved neuromorphic control for high speed divergence-based landings of mavs. IEEE Robotics and Automation Letters **5**(4), 6239–6246 (2020) 1
17. Hamaguchi, R., Furukawa, Y., Onishi, M., Sakurada, K.: Hierarchical neural memory network for low latency event processing. In: CVPR. pp. 22867–22876 (2023) 2, 4, 5, 7, 8, 10, 11, 13
18. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016) 7
19. Hidalgo-Carrió, J., Gehrig, D., Scaramuzza, D.: Learning monocular dense depth from events. In: 2020 International Conference on 3D Vision (3DV). pp. 534–542. IEEE (2020) 2, 11
20. Hu, Y., Delbruck, T., Liu, S.C.: Learning to exploit multiple vision modalities by using grafted networks. In: European Conference on Computer Vision. pp. 85–101. Springer (2020) 10
21. Kamal, U., Dash, S., Mukhopadhyay, S.: Associative memory augmented asynchronous spatiotemporal representation learning for event-based perception. In: The Eleventh International Conference on Learning Representations (2022) 2, 4
22. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 7
23. Lagorce, X., Orchard, G., Galluppi, F., Shi, B.E., Benosman, R.B.: Hots: a hierarchy of event-based time-surfaces for pattern recognition. IEEE transactions on pattern analysis and machine intelligence **39**(7), 1346–1359 (2016) 2, 3
24. Li, J., Li, J., Zhu, L., Xiang, X., Huang, T., Tian, Y.: Asynchronous spatio-temporal memory network for continuous event-based object detection. IEEE Transactions on Image Processing **31**, 2975–2987 (2022) 3, 10
25. Li, Y., Zhou, H., Yang, B., Zhang, Y., Cui, Z., Bao, H., Zhang, G.: Graph-based asynchronous event processing for rapid object recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 934–943 (2021) 2, 4
26. Liu, B., Xu, C., Yang, W., Yu, H., Yu, L.: Motion robust high-speed light-weighted object detection with event camera. IEEE Transactions on Instrumentation and Measurement (2023) 3, 10
27. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10012–10022 (2021) 7
28. Maqueda, A.I., Loquercio, A., Gallego, G., García, N., Scaramuzza, D.: 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. pp. 5419–5427 (2018) 4
29. Messikommer, N., Gehrig, D., Loquercio, A., Scaramuzza, D.: Event-based asynchronous sparse convolutional networks. In: European Conference on Computer Vision. pp. 415–431. Springer (2020) 4
30. Perot, E., De Tournemire, P., Nitti, D., Masci, J., Sironi, A.: Learning to detect objects with a 1 megapixel event camera. Advances in Neural Information Processing Systems **33**, 16639–16652 (2020) 10

31. Sabater, A., Montesano, L., Murillo, A.C.: Event transformer. a sparse-aware solution for efficient event data processing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2677–2686 (2022) 4
32. Schaefer, S., Gehrig, D., Scaramuzza, D.: Aegnn: Asynchronous event-based graph neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12371–12381 (2022) 2, 4, 10
33. Sekikawa, Y., Hara, K., Saito, H.: Eventnet: Asynchronous recursive event processing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3887–3896 (2019) 4
34. Sironi, A., Brambilla, M., Bourdis, N., Lagorce, X., Benosman, R.: Hats: Histograms of averaged time surfaces for robust event-based object classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1731–1740 (2018) 3
35. Son, B., Suh, Y., Kim, S., Jung, H., Kim, J.S., Shin, C., Park, K., Lee, K., Park, J., Woo, J., Roh, Y., Lee, H., Wang, Y., Ovsiannikov, I., Ryu, H.: 4.1 a 640×480 dynamic vision sensor with a 9µm pixel and 300meps address-event representation. In: 2017 IEEE International Solid-State Circuits Conference (ISSCC). pp. 66–67 (2017). https://doi.org/10.1109/ISSCC.2017.7870263 1
36. Sun, S., Cioffi, G., De Visser, C., Scaramuzza, D.: Autonomous quadrotor flight despite rotor failure with onboard vision sensors: Frames vs. events. IEEE Robotics and Automation Letters **6**(2), 580–587 (2021) 1
37. Sun, Z., Messikommer, N., Gehrig, D., Scaramuzza, D.: Ess: Learning event-based semantic segmentation from still images. In: European Conference on Computer Vision. pp. 341–357. Springer (2022) 10
38. Vemprala, S., Mian, S., Kapoor, A.: Representation learning for event-based visuomotor policies. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 4712–4724. Curran Associates, Inc. (2021) 4
39. Wang, Q., Zhang, Y., Yuan, J., Lu, Y.: Space-time event clouds for gesture recognition: From rgb cameras to event cameras. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 1826–1835. IEEE (2019) 2
40. Wu, Y., He, K.: Group normalization. In: Proceedings of the European conference on computer vision (ECCV). pp. 3–19 (2018) 7
41. Xiao, T., Liu, Y., Zhou, B., Jiang, Y., Sun, J.: Unified perceptual parsing for scene understanding. In: Proceedings of the European conference on computer vision (ECCV). pp. 418–434 (2018) 10
42. Zhu, A.Z., Thakur, D., Özaslan, T., Pfrommer, B., Kumar, V., Daniilidis, K.: The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. IEEE Robotics and Automation Letters **3**(3), 2032–2039 (2018) 1, 11