Hierarchical Neural Memory Network for Low Latency Event Processing

Ryuhei Hamaguchi[1], Yasutaka Furukawa[2], Masaki Onishi[1], and Ken Sakurada[1]

[1] National Institute of Advanced Industrial Science and Technology (AIST)

[2] Simon Fraser University

ryuhei.hamaguchi@aist.go.jp, furukawa@sfu.ca, onishi@ni.aist.go.jp, k.sakurada@aist.go.jp

Abstract

This paper proposes a low latency neural network architecture for event-based dense prediction tasks. Conventional architectures encode entire scene contents at a fixed rate regardless of their temporal characteristics. Instead, the proposed network encodes contents at a proper temporal scale depending on its movement speed. We achieve this by constructing temporal hierarchy using stacked latent memories that operate at different rates. Given low latency event steams, the multi-level memories gradually extract dynamic to static scene contents by propagating information from the fast to the slow memory modules. The architecture not only reduces the redundancy of conventional architectures but also exploits long-term dependencies. Furthermore, an attention-based event representation efficiently encodes sparse event streams into the memory cells. We conduct extensive evaluations on three event-based dense prediction tasks, where the proposed approach outperforms the existing methods on accuracy and latency, while demonstrating effective event and image fusion capabilities. The code is available at https://hamarh.github.io/hmnet/.

1. Introduction

Latency matters for many vision applications such as autonomous vehicles or UAVs, directly affecting their safety and reliability measures. Latency is also crucial for better user experience in time-sensitive vision applications such as augmented reality, where the latency of standard RGB cameras is insufficient. For example, a vehicle traveling 80km/h will move 74cm within a frame of a standard 30fps camera.

Event cameras have extremely low latency. Unlike standard vision cameras that capture the intensity of all pixels at a fixed rate, event cameras asynchronously record intensity changes of individual pixels. This unique principle leads to extremely low latency (microseconds) and high temporal resolution, along with many other advantages such as high dynamic range and low power consumption. With such at-
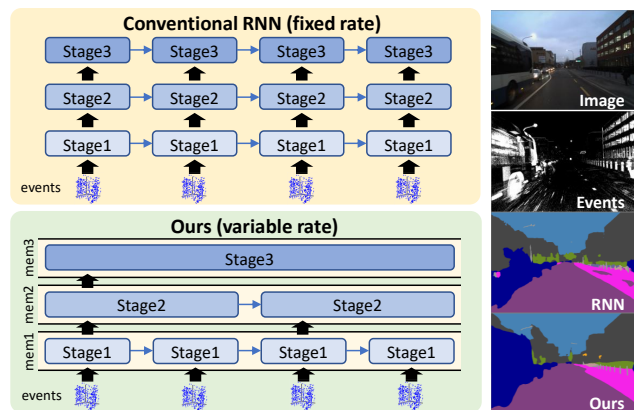

Figure 1. Schematic comparison between conventional and our methods. Our method adaptively processes dynamic and static scene contents using latent memories with variable rates. Our approach simultaneously processes fast-moving objects and static scene contents at low latency (See the right figure).

tractive characteristics, event cameras are becoming popular input devices for many dense prediction tasks such as semantic segmentation [1, 39], object detection [18, 30, 34], depth estimation [10, 26], and optical flow [11, 46].

Despite the emergence of low latency event-cameras, few works focused on low latency recognition models dedicated to event-based dense prediction tasks. Instead, most previous works apply standard CNN architectures equipped with the recurrent modules as a backbone [10, 15, 30, 39], resulting in the same latency levels as frame-based models. Another line of research applies Spiking Neural Networks (SNNs) for event data. However, SNNs suffer from low accuracy due to the lack of established training principles [40, 41] or high latency due to long simulation time steps [44]. We need backbone architectures that best exploit low latency and high temporal resolution of event data.

To this end, we propose a Hierarchical Neural Memory Network (HMNet) for low latency event processing. The key idea is to encode the scene contents at a proper temporal scale depending on their speed. For this purpose,

the proposed network builds multi-level latent memory with different operating rates (Fig. 1). The low-level memories run fast to quickly encode local and dynamic information, whereas high-level memories perform global reasoning on static information at a lower frequency. This design significantly reduces the computational loads in contrast to conventional methods that runs the entire forward path every time. The paper also proposes an Event Sparse Cross Attention (ESCA) that directly injects sparse event streams into dense memory cells with minimal information loss.

We conduct extensive evaluations on three event-based vision tasks (object detection, semantic segmentation, and monocular depth estimation) as well as event-image fusion task. Experimental results show that HMNet outperforms existing methods while reducing latency by 40%-50%.

2. Related Works

Event-based dense prediction often consists of the following two steps: 1) A construction of frame-based representations for events and 2) feature extraction using a backbone CNN followed by a task-specific head. The section reviews related works in these two steps, followed by discussion on specialized methods without these steps.

Event representation. An efficient and informative neural event representation is the key to effective event processing. Due to its sparse and asynchronous nature, event data are not directly applicable to many DNN models with dense-grid representations. The simplest solution is to convert events into histogram images [24, 28], but the method completely discards rich temporal information of events. In order to maximize the information, many event representations are proposed. Timestamp images [29] and Time Surface [6, 17] encode events based on the most recent timestamp at each pixel. HATS [38] further improved the methods by additionally encoding timestamps of past events. Voxel Grid [46] constructs spatiotemporal voxel representation by discretizing the time domain using a bilinear kernel.

Recently, more data-adaptive methods have been proposed [5, 9, 18, 21]. EST [9] generalized the above methods as Event Spike Tensor and further makes the kernel trainable end-to-end from data. Matrix-LSTM [5] applies pixel-wise LSTM to extract temporal features from the event stream. AED [21] and ASTMNet [18] improve Voxel Grid by adaptively adjusting kernel size and sampling position according to the event density.

In contrast, the proposed method is 1) Data-adaptive: the attention adaptively picks up necessary information from events based on the current memory state, 2) Robust to noise: the attention can filter out noisy events, and 3) High frame rate: the memory-based representation can work at a short interval (*e.g.* 5ms), contrary to the previous methods that require a long time (*e.g.* 50ms) to accumulate enough

data (excluding Time Surface and Matrix-LSTM).

Backbone architecture. A CNN architecture with a recurrent module is a popular architecture for event-based dense prediction tasks. Perot *et al.* [30] built a CNN architecture with stacked ConvLSTM [36] for object detection. Li *et al.* [18] also combined lite-weight LSTM with VGG16 [37]. The recurrent architectures based on U-Net [32] or stacked hourglass network [27] are also used for monocular depth estimation [10, 15] and semantic segmentation [39].

Several works proposed a backbone architecture for asynchronous event processing to better exploit the sparse and asynchronous nature of the event data. Messikommer *et al.* [25] proposed a sparse convolutional network for asynchronous event processing. Schaefer *et al.* [34] applied a graph convolutional neural network for event processing and proposed the efficient and asynchronous graph update rule for events. While these methods can dramatically reduce computational complexity, the performance is insufficient compared to conventional synchronous models.

Exploiting the sparsity of event data has been popular for classification tasks. EventNet [35] treated event data as a point cloud and applied MLPs event-by-event like PointNet [31]. By converting the trained MLPs into a lookup table, the network runs extremely fast at inference time. Event Transformer [19] also takes raw events as inputs and applies specialized transformer layers to extract spatiotemporal features for a group of events. Although these methods perform well on the classification task, applying them to dense prediction tasks is difficult since making dense predictions from the event-wise sparse features is not trivial.

Sabater *et al.* [33] proposed to update an internal memory using sparse patch-based representation extracted from the event Voxel Grid. This work is the most related to ours, but the method still relies on the conventional Voxel Grid representation, which is sub-optimal. Our method directly writes raw events into memory through cross-attention, which is more data-adaptive. Hierarchical Temporal Memory (HTM) [12] is conceptually related to our work. HTM consists of Spatial Pooling and Temporal Pooling that are conceptually similar to convolution and recurrent layers. HTM is more similar to RNNs and it operates at a single rate. In contrast, the proposed model operates at multiple rates, which is a key for low-latency processing.

3. Hierarchical Neural Memory Network

Our idea for low latency event processing is a multi-rate network architecture. A scene often contains objects with varying motion speeds. A network should run fast for high-speed motions, but conduct careful or global reasoning for slowly moving objects or scene context analysis.

To achieve this, the proposed network builds a temporal hierarchy using multi-level latent memories $\{z_1, ..., z_L\}$
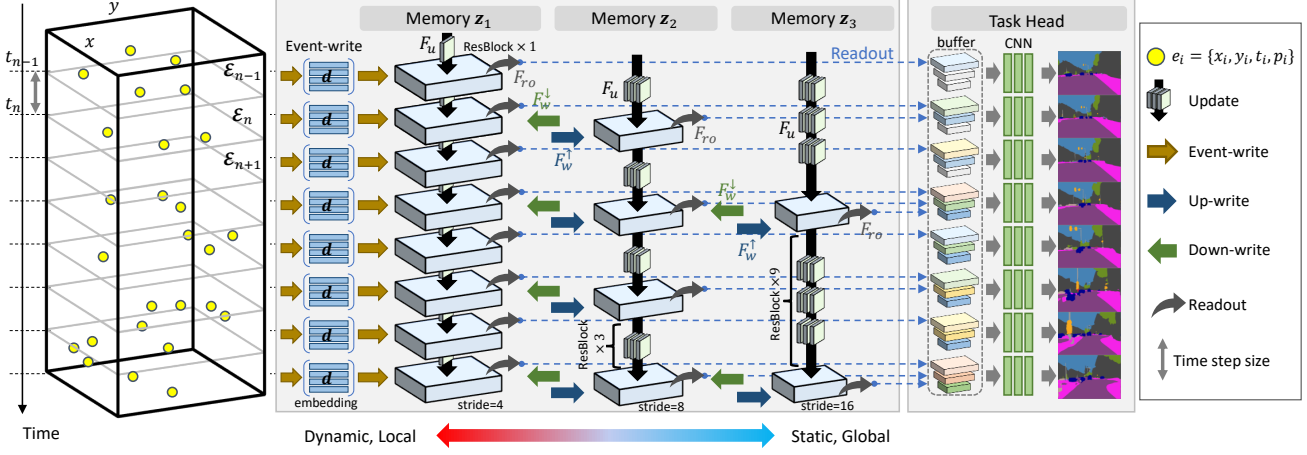
Figure 2. Illustration of Hierarchical Neural Memory Network. The network consists of a stack of multiple memories that operate at different rate. z_1 operates at highest rate and receives events at every time-steps ("event-write"). z_2, z_3 operate at lower rate, extracting deeper features during the cycle. Each memory exchange information with previous/next level memory ($F_w^\uparrow, F_w^\downarrow$), update its state with convolutional layers (F_u), and finally readout features for task head (F_{ro}).

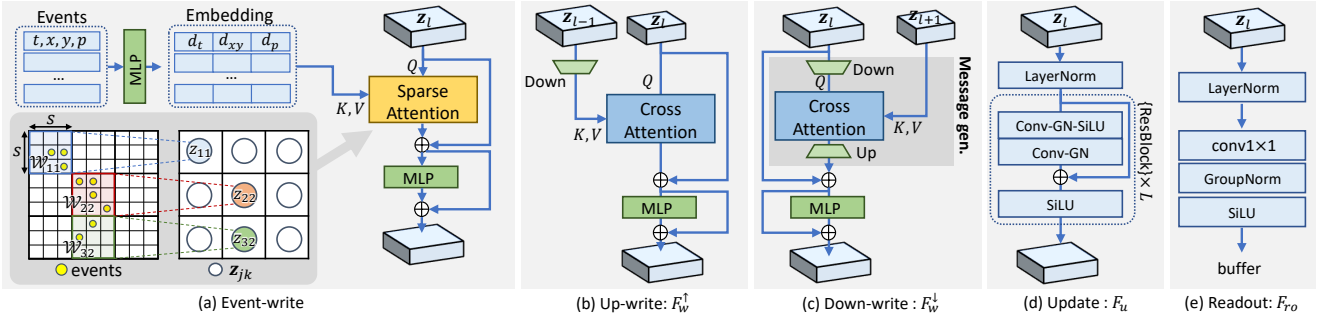| (a) Event-write | (b) Up-write : $F_w^\uparrow$ | (c) Down-write : $F_w^\downarrow$ | (d) Update : F_u | (e) Readout: F_{ro} |

Figure 3. Operations defined on latent memories. **Event-write** embeds events into the memory cells using Event Sparse Cross Attention. **Up/down-write** exchange features between memories by cross-attention. **Update** process internal state using residual layers. **Readout** computes features for the task head.

that operate in parallel at different rates. Fig. 2 shows an overview of the proposed network. The memories are stacked such that their operating rate decreases from z_1 to z_L. z_1 writes incoming events into its state (event-write) and quickly extracts local and dynamic information with a shallow network (update). The features are then propagated to higher memories (up-write) where global and static information is extracted with deeper networks (update). The network also has a top-down path (down-write) that enables low-level memories to exploit the contextual information to recognize dynamic motion accurately.

At the end of the operating cycle, each memory computes output features (readout) and puts them into a latent buffer. At every time step, the task head computes predictions from the features inside the latent buffer, exploiting low latency information and global context simultaneously.

Below, Sec. 3.1 explains the basic operations of latent memories. Sec. 3.2 proposes a method for writing raw event

data into the latent memory. Sec. 3.3 explains the overall operation flow and timing of the proposed network. Finally, Sec. 3.4 introduces the extension of the proposed network for multi-sensory inputs with varying frame rates.

3.1. Latent memory

We have a hierarchy of latent memory where memory cells update memories by attention mechanisms. Each memory has four operations: "*up-write*", "*down-write*", "*update*", and "*readout*". The memory z_1 has an additional operation, "*event-write*", to write raw event data into the memory state. The architectures for each operation are depicted in Fig. 3. Sec. 3.2 explains event-write in detail. Below, we explain the other four operations.

Up-write. The operation writes the memory state of the previous level z_{l-1} into current state z_l:
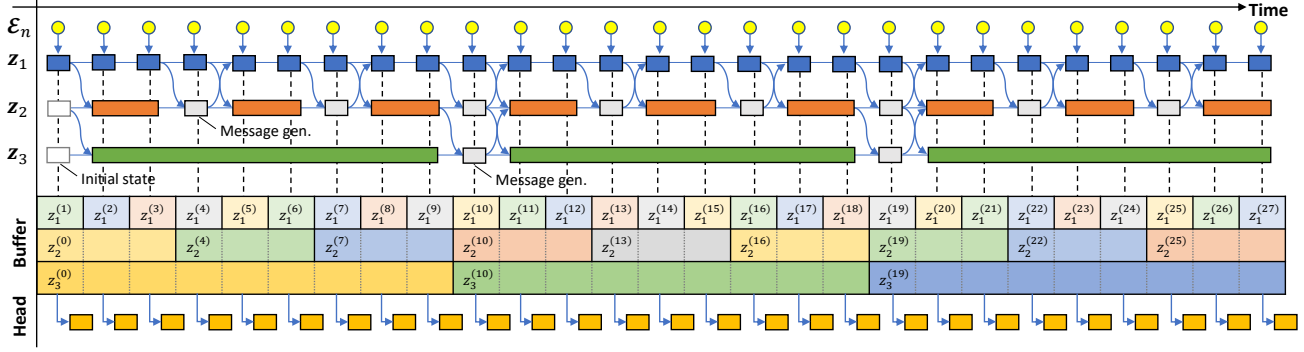
$$z_l \leftarrow F_w^\uparrow(z_l, z_{l-1}). \tag{1}$$

Figure 4. Timing chart of HMNet with three latent memories. z1 is the most time-sensitive, executing all the tasks (event-write, update, readout) in a single time step (blue blocks). z2 and z3 have a longer cycle and can execute more expensive update operations (orange and green blocks). In addition, they execute message and readout operations (gray blocks) at the end of the cycles. The features read out from the memories are put into a latent buffer and then fed to the task head for prediction (gold blocks).

To aggregate features from $\boldsymbol{z}_{l-1}$, the write function $F_w^\uparrow$ applies the window based multi-head cross-attention (W-MCA) that is our extension of W-MSA [22] for cross-attention:

$$\hat{\boldsymbol{z}}_l = \text{W-MCA}(\boldsymbol{z}_l, G_{down}(\boldsymbol{z}_{l-1})) + \boldsymbol{z}_l, \qquad (2)$$

$$\boldsymbol{z}_l \leftarrow \text{MLP}(\text{LN}(\hat{\boldsymbol{z}}_l)) + \hat{\boldsymbol{z}}_l, \qquad (3)$$

where LN is a Layer Normalization [2]. MLP consists of two conv 1×1 layers with a GELU activation function [14] in between. G_{down} is a strided convolutional block for downsampling.

Down-write. The operation writes the memory state of the next level $\boldsymbol{z}_{l+1}$ into current state $\boldsymbol{z}_l$:

$$\boldsymbol{z}_l \leftarrow F_w^\downarrow(\boldsymbol{z}_l, \boldsymbol{z}_{l+1}). \qquad (4)$$

$F_w^\downarrow$ is almost the same as Eq. 2, 3 except the down/up-sampling operation. $F_w^\downarrow$ applies G_{down} on $\boldsymbol{z}_l$ and upsamples the output of W-MCA with a bilinear upsampling layer (see Fig. 3b, 3c).

Update. The operation updates the internal state of the memories:

$$\boldsymbol{z}_l \leftarrow F_u(\text{LN}(\boldsymbol{z}_l)) \qquad (5)$$

The function F_u consists of several residual layers [13]. For high-level memories, a large number of layers can be used to encode high-level semantic information. On the other hand, for low-level memories, a few layers are used to prioritize latency. Specifically, we set the number of residual layers as $\{1, 3, 9\}$ for the memories $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3\}$.

Readout. The operation extracts an output feature $\boldsymbol{o}_l$ from the internal state of $\boldsymbol{z}_l$:

$$\boldsymbol{o}_l = F_{ro}(\text{LN}(\boldsymbol{z}_l)) \qquad (6)$$

The function F_{ro} consists of conv 1×1 followed by a Group Normalization [42] and a SiLU activation function [7].

3.2. Event write

Event cameras trigger an event asynchronously at each pixel whenever an accumulated brightness change exceeds a certain threshold. The output of the event cameras is a stream of asynchronous events $\mathcal{E} = \{\boldsymbol{e}_i\}_{i=1}^N$, where each event $\boldsymbol{e}_i = \{t_i, x_i, y_i, p_i\}$ encodes four values: the timestamp t_i, the pixel location (x_i, y_i), and the polarity $p_i \in \{1, -1\}$ that represent the direction of brightness change (increase or decrease).

To embed the sparse event stream into dense memory cells, we propose an Event Sparse Cross Attention (ESCA). It features high efficiency and robustness to noise by exploiting the sparsity of events through window-based attention and introducing a trainable "event gate" inside the attention computation.

At every time step t_n, the network receives events $\mathcal{E}_n = \{\boldsymbol{e}_i\}_{i=1}^N$ triggered during $[t_{n-1}, t_n]$ as inputs. Let $\boldsymbol{z}_{jk}$ be the memory cell at a spatial location (j, k) and s be the global stride of the memory cells; we define a $s \times s$ spatial window $\mathcal{W}_{jk}$ for each memory cell $\boldsymbol{z}_{jk}$. ESCA calculates cross attention between the memory cell $\boldsymbol{z}_{jk}$ and a subset of events $\mathcal{E}_n^{(j,k)}$ inside the window $\mathcal{W}_{jk}$ (Fig. 3a).

$$\boldsymbol{h}_{jk} = \text{CrossAttn}(\boldsymbol{z}_{jk}, \mathcal{E}_n^{(j,k)}), \qquad (7)$$

The attention calculation is efficient, because Eq. 7 is required only for the memory cells with events inside their local window. Below, we detail the attention calculation.

Event embedding. Each event $\boldsymbol{e}_i \in \mathcal{E}_n^{(j,k)}$ is first transformed into an embedding vector $\boldsymbol{d}_i$ using MLPs.

$$\boldsymbol{d}_t = E_t(\hat{t}_i), \quad \boldsymbol{d}_{xy} = E_{xy}(\hat{x}_i, \hat{y}_i), \quad \boldsymbol{d}_p = E_p(p_i),$$
$$\boldsymbol{d}_i = \text{LN}([\boldsymbol{d}_t, \boldsymbol{d}_{xy}, \boldsymbol{d}_p]), \qquad (8)$$

where E_t, E_{xy}, E_p are MLPs with two hidden layers and a Layer Normalization [2] in between, and $[,]$ is a concatenation operation. $(\hat{x}_i, \hat{y}_i)$ is a relative position of the event

e_i in the window $\mathcal{W}_{jk}$, and $\hat{t}_i$ is a relative timestamp in the time period $[t_{n-1}, t_n]$.

Sparse attention with event gate. The attention is then computed between the memory cell z_{jk} and the embedding vectors $\boldsymbol{d} \in \mathcal{R}^{L \times D}$ of all the L events in $\mathcal{E}_n^{(j,k)}$:

$$\boldsymbol{q}_{jk} = \mathcal{Q}(\boldsymbol{z}_{jk}), \quad \boldsymbol{K} = \mathcal{K}(\boldsymbol{d}), \quad \boldsymbol{V} = \mathcal{V}(\boldsymbol{d}),$$
$$\boldsymbol{h}_{jk} = \text{softmax}(\boldsymbol{q}_{jk}\boldsymbol{K}^T/\sqrt{D})\boldsymbol{V}. \tag{9}$$

where $\mathcal{Q}, \mathcal{K}, \mathcal{V}$ are the MLPs for computing query, key, and value. Since raw events are sparse and noisy, Eq. 9 assigns a large weight to the noisy events when there are a few or no true events inside the window $\mathcal{W}_{jk}$. We address the problem by concatenating a trainable parameter w, namely "event gate" before the softmax function:

$$\boldsymbol{a}_{jk} = \text{softmax}([\boldsymbol{q}_{jk}\boldsymbol{K}^T/\sqrt{D}, w]). \tag{10}$$

$\boldsymbol{a}_{jk}$ is the attention weights with a length of $(L+1)$. The last element of $\boldsymbol{a}_{jk}$ corresponds to the attention for the event gate that acts as a noise filter; Since noise will have a weak correlation with the current memory state, the softmax function will assign a high weight to the event gate rather than the noisy events. Thus, the event gate prevents the noise events from being written to the memory state.

Finally, the attention is calculated by omitting the last element of $\boldsymbol{a}_{jk}$ and multiplying values $\boldsymbol{V}$:

$$\boldsymbol{h}_{jk} = [\boldsymbol{a}_{jk}]_{1:L}\boldsymbol{V}. \tag{11}$$

3.3. Multi-level latent memory

Fig. 4 describes the overall operation flow. The network has three levels of memory states $\{z_1, z_2, z_3\}$ with an operating cycle of $\{1, 3, 9\}$ time-steps. Each memory runs on a different thread asynchronously except for the timing of feature transfer.

At every time-step, z_1 writes incoming event data and executes event-write, update, and readout operations during the time-step. The internal state of z_1 is then written to z_2 at every 3 time-steps. z_2 executes "write" and "update" operations during 2 time-steps, and then executes "readout" and message generateion at the last time-step. z_3 works similarly to z_2 but has the longest cycle of 9 time-steps. Note that the message generation of "down-write" (Fig. 3c) is executed at the higher memory thread, which saves computations at the time-sensitive lower memory thread.

The latent buffers will store output features that are read out from each memory. At every time-step, the task head utilizes the features inside the latent buffers to make a prediction, exploiting temporally multi-scale features.

3.4. Sensor fusion

Our architecture can naturally be extended to the multi-sensory inputs with varying operating frequencies. We

achieve this by simply setting the frequency of memory writes to that of the sensor operations. Specifically, for event-image fusion, the (RGB) image data is first embedded into a feature map using an image encoder. The feature map is then written to the high-level memory z_3 every 45ms in the same way as the "up-write" operation. The event data is similarly written to the low-level memory z_1 every 5ms.

4. Experiments

The section evaluates the accuracy and the latency of the HMNet on three tasks: semantic segmentation, object detection, and monocular depth estimation.

Below, Sec. 4.1 explains the basic settings of the experiments, and Sec. 4.2 to Sec. 4.4 show the results on each task. Finally, Sec. 4.5 performs the ablation study to verify the effectiveness of each component of the proposed model. See supplementary materials for more detailed setups.

4.1. Basic setups

HMNet variants. We build four variants of HMNet, HMNet-B1/L1/B3/L3. HMNet-B1 is the simplest form with only one latent memory. HMNet-B3 is its multi-level version with three latent levels. HMNet-L1/L3 is a high-capacity version of HMNet-B1/B3 with increased latent dimensions. Specifically, let D be the dimension of each memory and H be the number of the head for cross attention in the "write" operations; the configurations are as follows:

- HMNet-B1: $D = 128, H = 4$
- HMNet-L1: $D = 256, H = 8$
- HMNet-B3: $D = (128, 256, 256), H = (4, 8, 8)$
- HMNet-L3: $D = (256, 256, 256), H = (8, 8, 8)$

For the memories $\{z_1, z_2, z_3\}$, we set the global stride of features s as $\{4, 8, 16\}$ and the operation cycle as $\{1, 3, 9\}$ time-steps, where the time-step size is set as 5ms.

Baselines. We built baselines by 1) Using Time Surface to convert raw events into frame-based representations; 2) Taking a popular backbone network (ResNet, ConvNeXt, or Swin Transformer); and 3) Adding a task head to output predictions. We also built their recurrent version by inserting ConvGRU [3] between stages.

Event-image fusion. We extended HMNet-B3/L3 for event-image fusion as explained in Sec. 3.4. For the image encoder, the layers until the stage3 of ResNet-18 [13] are used. For comparison, we also extend the baselines by concatenating the most recent image frame with its inputs.

Training and inference. For training, we used Adam [16] and AdamW [23] as an optimizer, depending on the task. We set the batch size as 16 and the initial learning rate as 2.0e-4 with a cosine learning rate decay. We used random resize, crop, and horizontal flip for data augmentation. Unless otherwise mentioned, we test all the methods using
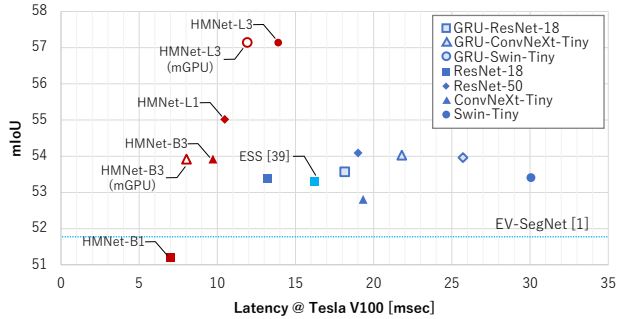
Figure 5. Results on DSEC-Semantic dataset. Lines depict the methods without latency value. All the methods are evaluated using only events. "mGPU" is a result obtained by multi-GPU inference.
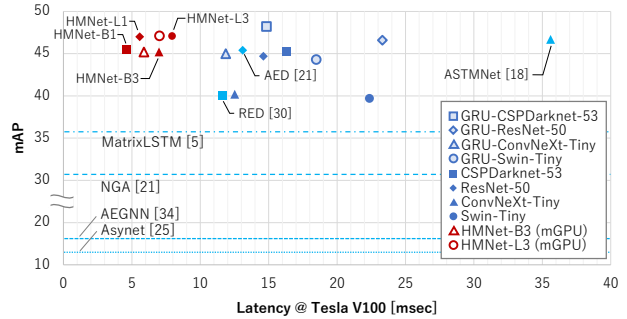

Figure 6. Results on GEN1 dataset. Lines depict the methods without latency value. The latency value of ASTMNet is borrowed from [18], where they used Titan Xp GPU. "mGPU" is a result obtained by multi-GPU inference.

Table 1. Resutls of event-RGB fusion on DSEC-Semantic dataset. The performances are measured in mIoU.

| | | Inputs | | |
	Backbone	event only	+left-RGB	+right-RGB
Baseline	ResNet50	54.1	50.6	48.9
HMNet (ours)	HMNet-B3	53.8	55.5	54.0
	HMNet-L3	55.0	57.4	57.4

Tesla V100 GPU and 20-core Intel(R) Xeon(R) Gold 6148 CPU @ 2.4GHz.

Justification for hyperparameter tuning. We manually determined the hyperparameters for all the experiments. To ensure that they are not overly tuned, we compared the manual tuning result to automatic tuning with Hyperopt [4] on the object detection task, where we observed similar accuracy (44.7 and 44.4 mAP for manual and automatic tuning).

4.2. Semantic segmentation

Setups. The experiments are conducted on DSEC-Semantic dataset [39]. The dataset contains street-scene event data with a resolution of 640×480 pixels. It also contains RGB frames recorded at 20Hz. Note that the cameras have different viewpoints; thus, the RGB frames are not perfectly aligned with the event data. The pseudo pixel-wise annotations are available at 20Hz. We used the task head of UPerNet [43] and trained our models for 60k iterations.

Results. Fig. 5 shows the results. HMNet-B3/L3 outperforms the state-of-the-art ESS model [39] on both accuracy and latency. Note that ESS utilizes additional training data on the RGB domain, while our methods are trained only on the event data. HMNet-L3 also outperforms the baselines with strong backbones. The effectiveness of HMNet-L3 is shown in a qualitative sample in Fig. 7 (top row). The comparison between HMNet-B1/L1 and HMNet-B3/L3 shows the performance gain obtained by stacking multiple memories (+3.8%/+3.2%). The latency of HMNet-B3/L3 can be

further reduced by parallel computation when multi-GPU is available (depicted by "mGPU" in the figure).

Event-image fusion. We trained the fusion models using events and frames from the left RGB camera. Table 1 compares the results in the case with and without RGB fusion. The baselines perform worse with RGB fusion due to the misalignment between events and RGB frames, whereas HMNet shows improved accuracy. Even with the misaligned inputs, cross-attention can robustly associate features, enabling the model to leverage both modalities. We also test the models using the images from the right RGB camera at inference time. To our surprise, the HMNet models keep the accuracy high, showing their robustness to the different sensor setups.

4.3. Object detection

Setups. The experiments are conducted on GEN1 dataset [30]. The dataset includes about 40 hours of event data with a resolution of 304×240 pixels. The bounding box annotations are available at 1Hz to 4Hz, depending on the video sequences. The labels are defined for two classes: pedestrian and car. We built a lite-weight detection head based on YOLOX [8]. Specifically, we adopt FPN [20] with additional bottom-up feature fusion instead of PAFPN [8].

Results. Fig. 6 compares the HMNet with the previous methods and baselines. The proposed method outperforms the state-of-the-art methods (ASTMNet [18] and AED [21]) in both accuracy and latency. Primarily, HMNet-B1/L1 performs remarkably well though they have only a single-level memory. For instance, HMNet-B1 performs competitively to AED [21] while reducing the latency by 57%, showing the effectiveness of the proposed ESCA. A qualitative sample in Fig. 7 (second row) shows the effectiveness of HMNet for detecting fast-moving objects.

Although the GRU-CSPDarknet-53 baseline achieved the best accuracy, the model has higher latency than ours
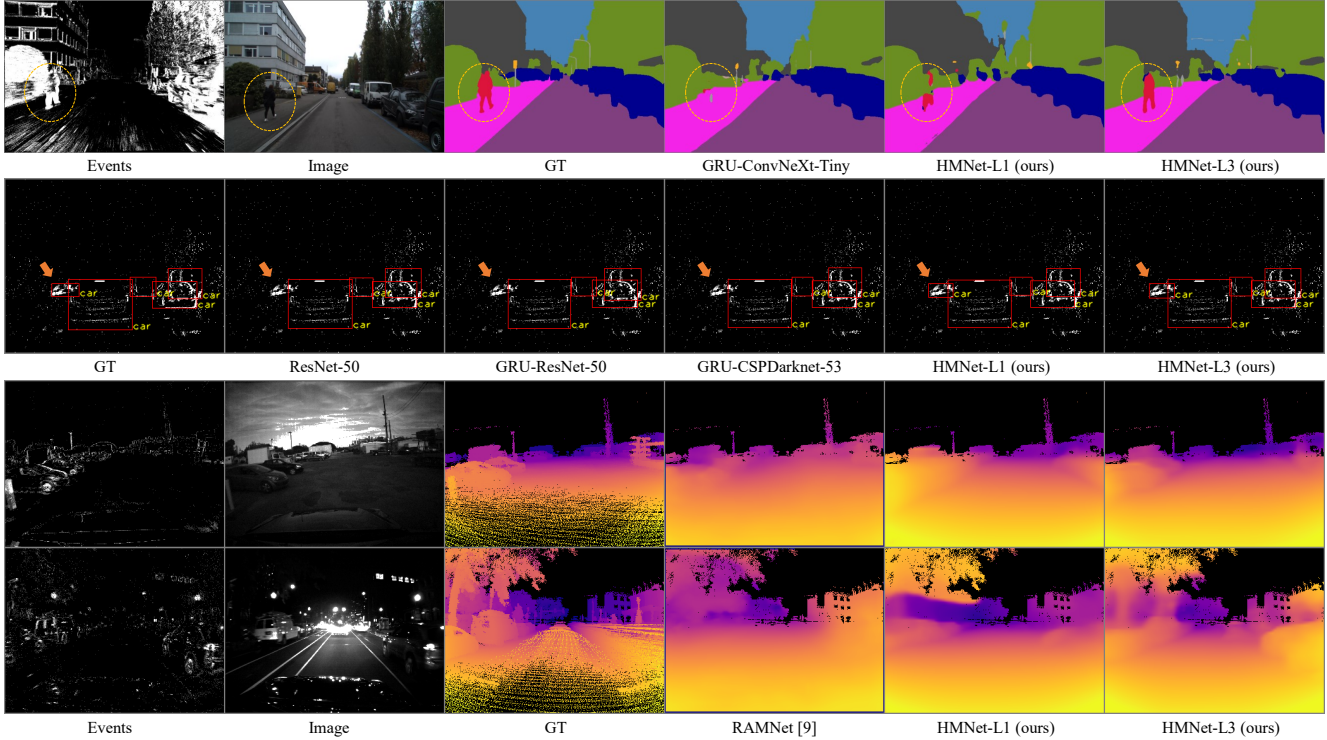
Figure 7. Qualitative results on each dataset. From top to bottom, the results for DSEC-Semantic, GEN1, and MVSEC *outdoor day1/night1* are shown, respectively. HMNet-L3 correctly segments the pedestrian (top row), detects a fast-moving car in the opposite lane (second row), and estimates a scene depth accurately around moving cars (third and fourth row). We cut out sky region for MVSEC results.

(more than double compared to HMNet-L1). More importantly, the recurrent baselines require a long accumulation time (*e.g.* 50ms) for constructing an event frame, which limits the frame rate of these methods.

4.4. Monocular depth estimation

Setups. Following Gehrig *et al*. [10], we pretrained our models on the synthetic Eventscape dataset [10]. We then fine-tuned and evaluated the models on real MVSEC dataset [45]. MVSEC dataset consists of street-scene event data and gray-scale images recorded by a DAVIS event camera with a resolution of 346×260 pixels. Since the DAVIS camera is coaxial, events and gray-scale images are aligned initially. The ground truth depth maps are recorded at 20Hz using a LiDAR sensor. Note that they are not synchronized with the gray-scale images. We used *outdoor day2* for training and *outdoor day1* and *outdoor night1* for evaluation.

Results. Table 2 shows the results on MVSEC dataset. On the daytime sequence, HMNet outperforms the previous methods while reducing latency (*e.g.* 44% reduction comparing HMNet-B3 and RAMNet [10]). Although the ResNet-50 baseline performs the best on night-time sequence, HMNet-B3 achieves competitive performance with much lower latency (65% reduction). On both sequences,

HMNet-B3/L3 is more accurate than HMNet-B1/L1, showing the effectiveness of the multi-level memory architecture. The qualitative results in Fig. 7 (bottom two rows) show that HMNet can accurately estimate the cars in the lanes.

Event-image fusion. Table 2 reports the results for event-image fusion. The baselines show degraded accuracy with the image fusion, which is due to the temporal mismatch between the images and the ground truth. The models need to fuse up-to-date events with an old image frame at inference timing, which is difficult to handle by naive concatenation. On the other hand, the HMNet has dedicated architecture to handle such temporal variation and hence enjoys performance gain with the image fusion on the daytime sequence. Meanwhile, the HMNet models perform worse with image fusion on the night-time sequence due to the image appearance gap between daytime and night-time.

4.5. Ablation study

We conducted ablation studies on several key elements of HMNet using DSEC-Semantic and GEN1 datasets.

Effect of event gate and down-write. Table 3 left investigates the effect of the event gate used in ESCA and the down-write operation of HMNet. The results show the effectiveness of the components. Applying both of them im-

Table 2. Comparison of performances on MVSEC dataset. The top three results are colored in orange, green, and blue.

	Backbone	Input	outdoor day1						outdoor night1						Latency [ms]
			$\delta1\uparrow$	$\delta2\uparrow$	$\delta3\uparrow$	REL↓	RMS↓	RMSlog↓	$\delta1\uparrow$	$\delta2\uparrow$	$\delta3\uparrow$	REL↓	RMS↓	RMSlog↓	
E2Depth [15]	ConvLSTM	event	0.567	0.772	0.876	0.346	8.564	0.421	0.408	0.615	0.754	0.591	11.210	0.646	-
RAMNet [10]	ConvGRU	event+RGB	0.541	0.778	0.877	0.303	8.526	0.424	0.296	0.502	0.635	0.583	13.340	0.830	9.0
Baseline	ResNet-18	event	0.689	0.837	0.921	0.267	6.981	0.334	0.533	0.725	0.850	0.321	8.893	0.451	8.9
	ResNet-18	event+RGB	0.679	0.830	0.915	0.264	7.329	0.352	0.472	0.651	0.771	0.352	11.506	0.576	8.4
	ResNet-50	event	0.683	0.834	0.921	0.267	7.064	0.333	0.535	0.726	0.850	0.331	8.817	0.449	14.7
	ResNet-50	event+RGB	0.677	0.826	0.912	0.265	7.485	0.357	0.474	0.678	0.802	0.335	10.624	0.526	14.4
Baseline-GRU	ResNet-18	event	0.666	0.814	0.903	0.299	7.685	0.367	0.534	0.740	0.868	0.330	8.356	0.436	8.5
	ResNet-18	event+RGB	0.493	0.675	0.796	0.345	8.916	0.520	0.327	0.553	0.703	0.385	11.547	0.678	8.8
HMNet (ours)	HMNet-B1	event	0.496	0.702	0.821	0.385	9.088	0.509	0.504	0.702	0.827	0.387	8.875	0.505	2.4
	HMNet-L1	event	0.625	0.798	0.893	0.310	8.383	0.393	0.497	0.679	0.796	0.434	9.406	0.561	4.1
	HMNet-B3	event	0.684	0.839	0.924	0.270	7.101	0.332	0.523	0.722	0.846	0.323	8.935	0.462	5.0 (4.1)
	HMNet-B3	event+RGB	0.707	0.857	0.937	0.252	6.972	0.318	0.512	0.714	0.836	0.316	10.089	0.476	5.4 (4.1)
	HMNet-L3	event	0.690	0.849	0.930	0.254	6.890	0.319	0.513	0.714	0.837	0.323	9.008	0.482	6.9 (5.4)
	HMNet-L3	event+RGB	0.717	0.868	0.940	0.230	6.922	0.310	0.497	0.661	0.784	0.349	10.818	0.543	7.1 (5.4)

Table 3. Ablation study on the event gate and the down-write (left) and the cycle length of the latent memories (right).

event gate	down-write	DSEC-Semantic mIoU	GEN1 mAP
		47.5	41.6
✓		47.8	42.4
	✓	48.6	41.7
✓	✓	49.1	42.7

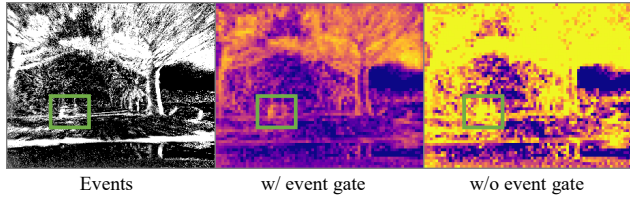| Cycle length z_3	DSEC-Semantic mIoU	GEN1 mAP
12	49.7	42.3
9	49.1	42.7
6	47.6	42.2
3	48.2	42.3

Figure 8. Visualization of attention weights of ESCA. A pixel with yellow color shows a memory cell receiving a large attention weight.
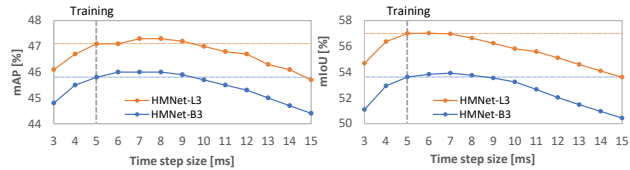
Events   w/ event gate   w/o event gate

Figure 9. Sensitivity analysis on the time step size at inference time. Left: DSEC-Semantic dataset. Right: GEN1 dataset.

proves the accuracy by +1.6% and +1.1% for each dataset. To further analyze the effect of the event gate, we visualized the attention weights of each memory cell at z_1. Specifically, Fig. 8 visualizes the accumulated attention weights written into each cell during 10 time steps (*i.e.* 50ms). Without the event gate, the latent memory receives much noisy information. In contrast, with the event gate, the latent memory selectively picks up essential information about the scene (*e.g.* the car in the green rectangle).

Sensitivity analysis on time step size. In practice, the time step size needs to be adjustable depending on the available hardware at inference time. Hence, we performed the sensitivity analysis on the time step size. We trained HMNet-B3/L3 using a time step size of 5ms and evaluated them with various step sizes (3ms to 15ms). Fig. 9 shows the analysis. The models perform well until the time step size reaches 10ms even showing better accuracy around 6ms-8ms. After that, the accuracy degrades linearly.

Sensitivity analysis on cycle length. Table 3 right shows the accuracy of HMNet-B3 when the cycle length of z_3 is changed between 3 to 12. The cycle length of 9 performs better than the shorter ones, showing the importance of long-term information propagation by z_3. Though, too long cycle length impairs the accuracy on GEN1 dataset.

5. Conclusion

This paper proposed a novel Hierarchical Neural Memory Network (HMNet) that builds a temporal hierarchy with multi-level memories for low latency event processing. The paper also proposed an Event Sparse Cross Attention (ESCA) for embedding sparse event streams into the latent memory of HMNet with minimal information loss. The experimental results showed that the HMNet can run faster than previous methods while achieving competitive or even better accuracy. The experiments on event-image fusion further showed the effectiveness of HMNet for fusing temporally unaligned sensory inputs. Our method is not limited to event data. We plan to extend our method to a variety of sensors (*e.g.* RGB, Lider, or RGB-Lidar fusion) and broader vision applications, such as video recognition, action recognition, and pose estimation.

Acknowledgement

References

[1] Inigo Alonso and Ana C. Murillo. Ev-segnet: Semantic segmentation for event-based cameras. In *CVPRW*, 2019. 1

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. In *CoRR*, 2016. 4

[3] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. In *ICLR*, 2016. 5

[4] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *ICML*, 2013. 6

[5] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. A differentiable recurrent surface for asynchronous event-based data. In *ECCV*, 2020. 2

[6] Haosheng Chen, David Suter, Qiangqiang Wu, and Hanzi Wang. End-to-end learning of object motion estimation from retinal events for event-based object tracking. In *AAAI*, 2020. 2

[7] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107, 2018. 4

[8] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. In *CoRR*, 2021. 6

[9] Daniel Gehrig, Antonio Loquercio, Konstantinos Derpanis, and Davide Scaramuzza. End-to-end learning of representations for asynchronous event-based data. In *ICCV*, 2019. 2

[10] Daniel Gehrig, Michelle Ruegg, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Combining events and frames using recurrent asynchronous multimodal networks for monocular depth prediction. *IEEE Robotics and Automation Letters*, 6, 2021. 1, 2, 7, 8

[11] Mathias Gehrig, Mario Millhausler, Daniel Gehrig, and Davide Scaramuzza. E-raft: Dense optical flow from event cameras. In *3DV*, 2021. 1

[12] Jeff Hawkins. Hierarchical Temporal Memory including HTM Cortical Learning Algorithms. Technical report, Numenta, 2011. 2

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 4, 5

[14] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). In *CoRR*, 2016. 4

[15] Javier Hidalgo-Carrió, Daniel Gehrig, and Davide Scaramuzza. Learning monocular dense depth from events. In *3DV*, 2020. 1, 2, 8

[16] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[17] 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 TPAMI*, 39, 2017. 2

[18] Jianing Li, Jia Li, Lin Zhu, Xijie Xiang, Tiejun Huang, and Yonghong Tian. Asynchronous spatio-temporal memory network for continuous event-based object detection. *IEEE Transactions on Image Processing*, 31, 2022. 1, 2, 6

[19] Zhihao Li, M. Salman Asif, and Zhan Ma. Event transformer. In *CoRR*, 2022. 2

[20] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 6

[21] Bingde Liu. Motion robust high-speed light-weighted object detection with event camera. In *CoRR*, 2022. 2, 6

[22] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 4

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 5

[24] Ana I. Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso Garcia, and Davide Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *CVPR*, 2018. 2

[25] Nico Messikommer, Daniel Gehrig, Antonio Loquercio, and Davide Scaramuzza. Event-based asynchronous sparse convolutional networks. In *ECCV*, 2020. 2

[26] Yeongwoo Nam, Mohammad Mostafavi, Kuk-Jin Yoon, and Jonghyun Choi. Stereo depth from events cameras: Concentrate and focus on the future. In *CVPR*, 2022. 1

[27] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016. 2

[28] Anh Nguyen, Thanh Toan Do, Darwin G. Caldwell, and Nikos G. Tsagarakis. Real-time 6dof pose relocalization for event cameras with stacked spatial lstm networks. In *CVPRW*, 2019. 2

[29] Paul K.J. Park, Baek Hwan Cho, Jin Man Park, Kyoobin Lee, Ha Young Kim, Hyo Ah Kang, Hyun Goo Lee, Jooyeon Woo, Yohan Roh, Won Jo Lee, Chang Woo Shin, Qiang Wang, and Hyunsurk Ryu. Performance improvement of deep learning based gesture recognition using spatiotemporal demosaicing technique. In *Proceedings - International Conference on Image Processing, ICIP*, 2016. 2

[30] Etienne Perot, Pierre de Tournemire, Davide Nitti, Jonathan Masci, and Amos Sironi. Learning to detect objects with a 1 megapixel event camera. In *NeurIPS*, 2020. 1, 2, 6

[31] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 2

[33] Alberto Sabater, Luis Montesano, and Ana C. Murillo. Event transformer. a sparse-aware solution for efficient event data processing. In *CVPRW*, 2022. 2

[34] Simon Schaefer, Daniel Gehrig, and Davide Scaramuzza. Aegnn: Asynchronous event-based graph neural networks. In *CVPR*, 2022. 1, 2

[35] Yusuke Sekikawa, Kosuke Hara, and Hideo Saito. Eventnet: Asynchronous recursive event processing. In *CVPR*, 2019. 2

[36] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, Wang-Chun Woo, and Hong Kong Observatory. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NeurIPS*, 2015. 2

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2

[38] 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*, 2018. 2

[39] Zhaoning Sun, Nico Messikommer, Daniel Gehrig, and Davide Scaramuzza. Ess: Learning event-based semantic segmentation from still images. In *ECCV*, 2022. 1, 2, 6

[40] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothee Masquelier, and Anthony S. Maida. Deep learning in spiking neural networks. *Neural Networks*, 111, 2018. 1

[41] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 12, 2018. 1

[42] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018. 4

[43] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018. 6

[44] Zhanglu Yan, Jun Zhou, and Weng Fai Wong. Near lossless transfer learning for spiking neural networks. In *AAAI*, 2021. 1

[45] Alex Zihao Zhu, Dinesh Thakur, Tolga Özaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The multi-vehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 3, 2018. 7

[46] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised event-based learning of optical flow, depth, and egomotion. In *CVPR*, 2019. 1, 2