This paper has been accepted for publication at the
European Conference on Computer Vision (ECCV), 2026

Low-latency Event-based Object Detection with Spatially-Sparse Linear Attention

Haiqing Hao[1], Zhipeng Sui[1], Rong Zou[2], Zijia Dai[3], Nikola Zubić[2], Davide Scaramuzza[2], and Wenhui Wang[1⋆]

[1] State Key Laboratory of Precision Measurement Technology and Instruments, Department of Precision Instrument, Tsinghua University, Beijing, China
[2] Robotics and Perception Group, University of Zurich, Zurich, Switzerland
[3] ShanghaiTech University, Shanghai, China

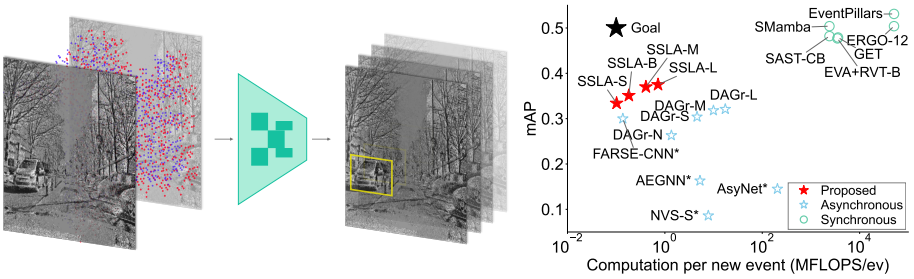
Abstract. Event cameras provide sequential visual data with spatial sparsity and high temporal resolution, making them attractive for low-latency object detection. Existing asynchronous event-based neural networks exploit this low-latency advantage by updating predictions event by event, but still suffer from two bottlenecks: recurrent architectures are difficult to train efficiently on long sequences, and improving accuracy often increases per-event computation and latency. Linear attention is appealing because it enables parallel training and recurrent inference. However, its dense state updates make per-event computation scale with the state size, yielding a poor accuracy-efficiency trade-off for object detection, where accurate localization requires fine-grained spatial states. The key challenge is therefore to introduce *sparse state activation* that exploits the spatial sparsity of events while preserving *efficient parallel training*. We propose Spatially-Sparse Linear Attention (SSLA), which introduces a mixture-of-spaces state decomposition and a scatter-compute-gather training procedure, enabling state-level sparsity as well as training parallelism. Building on SSLA, we develop an end-to-end asynchronous linear attention model, SSLA-Det, for low-latency event-based object detection. On Gen1 and N-Caltech101, SSLA-Det achieves state-of-the-art accuracy among asynchronous methods, reaching 0.375 mAP and 0.515 mAP, respectively, while reducing per-event computation by over 20× compared with the strongest prior asynchronous baseline, demonstrating the potential of linear attention for low-latency event-based vision. Code is available at: `https://github.com/haohq19/ssla`.

Keywords: Event camera · Linear attention · Object detection

1 Introduction

Event cameras provide sequential visual data with spatial sparsity and high temporal resolution, making them highly promising for low-latency perception [9,11,29]. Asynchronous event-based neural networks realize this potential by updating their predictions every time a new event arrives [36,37]. This event-driven

⋆ Corresponding author: `wwh@tsinghua.edu.cn`



(a) Events **(b)** SSLA **(c)** Detection **(d)** mAP vs. FLOPS

Fig. 1: Our method processes (a) asynchronous event sequence with (b) a sparsely activated linear attention neural network for (c) low-latency event-based object detection. On the Gen1 dataset, our SSLA-Det models achieve SOTA asynchronous mAP and lower FLOPS compared with previous asynchronous baselines (d). * refers to AP_{50}.

processing paradigm is particularly appealing for object detection in latency-critical scenarios, such as autonomous driving [11], drone obstacle avoidance [7], and vision-based control [15].

Despite their much lower latency, existing asynchronous event-based neural networks still lag behind their synchronous counterparts in accuracy [33, 49, 50]. The gap stems from two coupled architectural bottlenecks. The first is the parallel-recurrent bottleneck: the event-by-event inference paradigm naturally relies on recurrent architectures, whereas efficient training on long event sequences requires parallelization along the sequence dimension [13, 39]. The second is the accuracy-efficiency trade-off: improving accuracy typically requires larger and deeper models, while scaling the model increases per-event computation and consequently latency. A natural way to mitigate this trade-off is to exploit the spatial sparsity of event camera data through sparse neural network activation [36, 37]. Nevertheless, as receptive fields expand layer by layer in deep networks, sparse inputs can still induce dense activations. To preserve sparsity in deep layers and reduce computation, prior work has designed specialized architectures [11, 36], but this comes with additional architectural constraints that further limit accuracy.

Linear attention[4] has emerged as a promising asynchronous event-based model architecture since it naturally addresses the parallel-recurrent bottleneck [12, 20, 31, 46, 47]. However, existing methods are limited to relatively simple global-level classification tasks [38, 41], while more challenging local-level tasks, like object detection, remain unexplored. The main obstacle is the poor accuracy-efficiency trade-off due to the lack of state-level sparsity, *i.e.*, linear attention updates all elements of its state, making per-event computation scale with the state size. This is problematic for object detection, where accurate localization

[4] For convenience, here we use linear attention as a shorthand for parallel-trainable linear recurrent models, including state space models (SSMs) and linear recurrent neural networks (linear RNNs).

requires fine-grained spatial representations and therefore a large state size [13]. Although sparsifying state activation is conceptually straightforward, the key challenge is to maintain parallel training while gaining sparsity.

We address this challenge by introducing Spatially-Sparse Linear Attention (SSLA), a linear attention module with state-level sparsity while preserving its parallel training advantages. To enable state-level sparsity, we introduce a *mixture-of-spaces* (MOS) structure inspired by [6] that decomposes the global state into substates with spatially overlapping receptive fields, and each event activates only a few substates based on its location. To preserve the sparsity in deep networks, we aggregate the activations of each event from all its activated substates, preventing the expansion of the activated region. We further propose a *position-aware projection* (PAP) that projects events based on their relative positions within every activated substate, injecting state-relative spatial priors. We derive a *scatter-compute-gather* training procedure that parallelizes this sparse activation structure by sequence-level reorganization. Specifically, events are scattered into state-specific subsequences, computed in parallel by linear attention, and then gathered back into the original event sequence. In this way, SSLA makes linear attention sparse in space, recurrent in time, and parallel in training.

Building on the SSLA module, we present SSLA-Det, to the best of our knowledge, the first end-to-end asynchronous linear attention model for low-latency event-based object detection (Fig. 1 (a)-(c)). Experiments on Gen1 and N-Caltech101 show that SSLA-Det achieves a substantially improved accuracy-efficiency trade-off over prior asynchronous methods (Fig. 1 (d)), including state-of-the-art (SOTA) asynchronous mAP (0.375 on Gen1 and 0.515 on N-Caltech101) at much lower computational cost (over 20× reduction compared with previous SOTA [11]). Our contributions are as follows:

- We propose an SSLA module for sequential event modeling, including a MOS structure for state-level sparsity, PAP for spatial prior encoding, and a scatter-compute-gather procedure for efficient parallel training.
- We present SSLA-Det, to the best of our knowledge, the first end-to-end asynchronous linear attention model for event-based object detection.
- SSLA-Det sets a new accuracy-efficiency frontier, reaching 0.375 mAP on Gen1 and 0.515 mAP on N-Caltech101, while reducing computation by over 20× compared with the prior asynchronous SOTA method.

2 Related Work

2.1 Asynchronous Event-based Neural Networks

Asynchronous event-based neural networks treat event camera data as different geometric structures to leverage their spatial sparsity. This strategy includes graphs [3, 4, 11, 22, 37], submanifolds [25, 36], point clouds [39, 43], and sequences [13,19,38,41]. Graph-based methods [2–4,11,22,37] transform events into sparsely connected spatial-temporal graphs, and derive local update rules on the graph for

recurrent inference. However, they have limitations in temporal accumulation [4], failing to handle long event sequences. Submanifold methods [25, 36] assume that events lie on a spatial submanifold, and conduct convolution only on the submanifold to keep sparsity. However, this submanifold assumption does not strictly hold, and thus leads to suboptimal performance. Point cloud methods [39, 43] represent events as spatial-temporal 3-D point clouds, and process with PointNet [35]. This approach is limited to shallow neural networks, and thus has difficulty in challenging tasks. Sequence-based methods use causal sequence-to-sequence models for event processing including softmax attention or linear recurrent models. For example, [19] uses attention to process batched events at each timestep. EventSSM [38] and S7 [41] use SSMs, which enable parallel training and recurrent inference, but are limited to global-level classification tasks. EVA [13] explores linear attention for event-based object detection but still demands a dense backbone, while our method is fully end-to-end asynchronous.

2.2 State-level Sparsity in Linear Attention

A recent direction to improve linear attention is to introduce state-level sparsity. Mixture-of-Memories [6] and Sparse State Expansion [28] maintain multiple independent states and use a learned router to send each token to only a few memories. Our SSLA follows the same sparse state activation idea but is fundamentally different in that the construction of substates is spatially structured, which enables geometric routing of event embeddings and admits the position-aware projection that encodes spatial inductive bias. Concurrent work [40] also introduces local states in linear attention for local-level event-based vision, but requires spatial contraction at discrete timestamps, which is not asynchronously, event-by-event trainable. Our work, instead, uses a scatter-compute-gather algorithm to reorganize events into subsequences and does not rely on spatial contraction, which is fully event-by-event asynchronous.

3 Method

3.1 Problem Formulation: Asynchronous Event Processing

Event camera data are represented as a sequence of events $\mathcal{E} = \{e_i\}_{i=1}^{L}$, where each event $e_i = (\mathbf{x}_i, t_i, p_i)$ carries its spatial coordinates $\mathbf{x}_i \in \mathbb{R}^2$, a timestamp $t_i \in \mathbb{R}$ and a polarity $p_i \in \{+1, -1\}$. The sequence is temporally ordered so that $t_i \leq t_{i+1}$. Asynchronous event processing learns a causal stateful neural network $\mathcal{M}$ that takes $\mathcal{E}$ as input and makes predictions incrementally, which can be formulated as a causal sequence-to-sequence problem from $\{e_i\}_{i=1}^{L}$ to $\{\hat{\mathbf{y}}_i\}_{i=1}^{L}$. Specifically, for each new incoming event e_i, the model updates its state $\mathbf{S}_i$ and produces a new prediction $\hat{\mathbf{y}}_i$, as $(\hat{\mathbf{y}}_i, \mathbf{S}_i) = \mathcal{M}(e_i, \mathbf{S}_{i-1})$.

3.2 Preliminaries: Linear Attention

Linear attention models (linear RNNs, SSMs) are linear-time alternatives to softmax attention [44] for sequence-to-sequence modeling. Causal linear atten-

tion has an equivalent parallel and recurrent form, which enables both parallel training and recurrent inference [20]. Given an input sequence of embeddings $\{\mathbf{z}_i\}_{i=1}^{L}$, a causal linear attention module updates its hidden state $\mathbf{S}_i$ and computes outputs $\{\mathbf{o}_i\}_{i=1}^{L}$ as

$$\begin{aligned} \mathbf{S}_i &= g(\mathbf{z}_i) \odot \mathbf{S}_{i-1} + \phi(\mathbf{z}_i), \\ \mathbf{o}_i &= \rho(\mathbf{z}_i, \mathbf{S}_i), \end{aligned} \tag{1}$$

where g, ϕ, ρ are learnable projections (gating, updating, and output) and $\odot$ is the Hadamard product. This recurrent linear attention is parallel trainable with parallel scan [12] or chunk-wise algorithms [46], which is training-efficient on long event sequences. For convenience, we denote the map of linear attention as $\textbf{LinearAttention} : \{\mathbf{z}_i\}_{i=1}^{L} \mapsto \{\mathbf{o}_i\}_{i=1}^{L}$ in this paper.

3.3 Spatially-Sparse Linear Attention for Event Sequence Modeling

We introduce the Spatially-Sparse Linear Attention (SSLA) module for asynchronous event processing, which is illustrated in Fig. 2. The SSLA module takes an event sequence $\mathcal{E}$ with embeddings $\mathbf{v}_i \in \mathbb{R}^{D_{in}}$ as input, and outputs $\mathcal{E}$ with updated embeddings $\mathbf{o}_i \in \mathbb{R}^{D_{out}}$. To exploit the spatial sparsity of events, we first introduce a *mixture-of-spaces* (MOS) decomposition of the hidden state, which enables state-level spatial sparsity. We incorporate a *position-aware projection* in the SSLA module to encode spatial priors into event embeddings. To preserve the training efficiency of linear attention, we derive a *scatter-compute-gather* procedure for parallel training by reorganizing events to independent subsequences.

Sparse State Activation with Mixture-of-Spaces. We define Ω as the spatial domain of the event camera data, and decompose it into spatially local and overlapping patches, with each patch maintaining its own state independently. We construct these patches by applying a sliding window of size $P \times P$ with a stride of 1 over Ω. Let $\mathcal{P} = \{1, \ldots, K\}$ denote the indices of these patches, where each patch $k \in \mathcal{P}$ covering a spatial region $\mathcal{R}_k \subset \Omega$ maintains state $\mathbf{S}_k$.

For e_i at $\mathbf{x}_i$, we activate only specific patches that contain $\mathbf{x}_i$ for sparse state activation. We define the set of active patches of e_i as

$$\mathcal{K}_i = \{k \in \mathcal{P} \mid \mathbf{x}_i \in \mathcal{R}_k\}. \tag{2}$$

We pad the image domain to ensure that each event activates a constant number of states, *i.e.*, $|\mathcal{K}_i| = P^2 \triangleq A$. These A states are updated by Eq. (1) with event embeddings, and generate interim outputs $\{\mathbf{o}_{i,k} \mid k \in \mathcal{K}_i\}$. All active patches share the same linear attention parameters but maintain independent states. We aggregate the interim outputs as the updated embedding from all activated patches:

$$\mathbf{o}_i = \sum_{k \in \mathcal{K}_i} \mathbf{o}_{i,k}. \tag{3}$$

After aggregation, the embedding sequence length is unchanged, which preserves sparsity in deep layers.

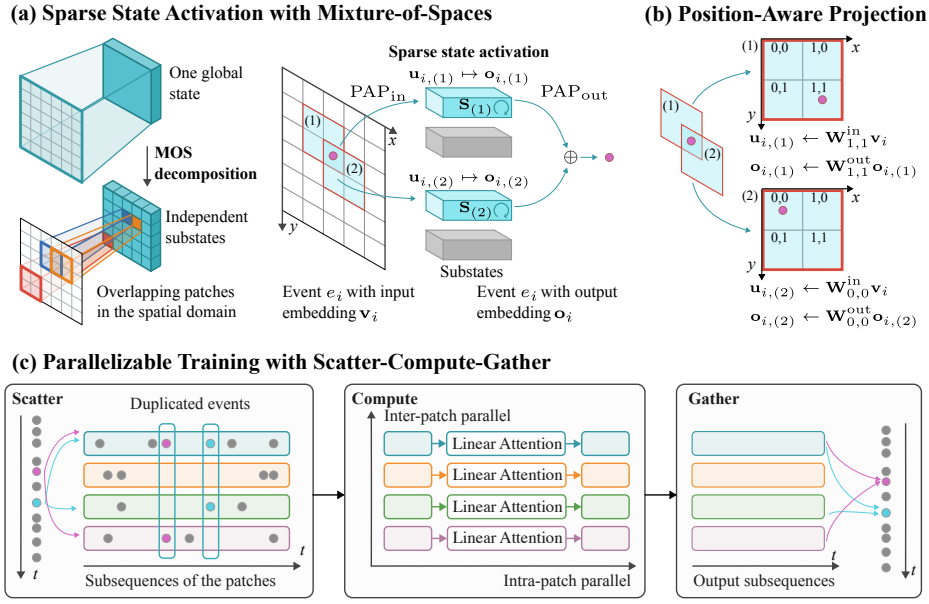


Fig. 2: Overview of the Spatially-Sparse Linear Attention (SSLA) module. (a) The global state is decomposed into independent substates, one per overlapping patch in the spatial domain. (1) and (2) are 2×2 patch examples that cover the event e_i (for brevity, we omit the other patches covering e_i). Their states are updated with the embedding of e_i, and the output embedding of e_i is the summation of the interim outputs from all the patches covering e_i. (b) In each patch, the embeddings are projected based on their relative position within the patch. (c) During training, with an event sequence as input, the events are scattered into patch-wise subsequences (duplicating each event across all the patches covering it), computed in parallel by a weight-shared linear attention, then gathered back to the original event order.

Position-Aware Projection. Sharing identical embeddings $\mathbf{v}_i$ for one event in all its activated patches ignores the spatial prior of the event, since the event is located at different relative positions in different patches. We therefore introduce a position-aware projection (PAP) of the embedding, which projects the embedding based on the event's relative positions inside the patches.

For an event at $\mathbf{x}_i$ in patch k with top-left global coordinates $\mathbf{c}_k \in \Omega$, we compute its relative position in the patch as

$$\boldsymbol{\delta}_{i,k} = \mathbf{x}_i - \mathbf{c}_k, \quad \text{where } \boldsymbol{\delta}_{i,k} \in \{0, ..., P-1\}^2. \tag{4}$$

We define the PAP as a linear transform by $\mathbf{W}^{\text{in}} \in \mathbb{R}^{P \times P \times D_{out} \times D_{in}}$ and $\mathbf{W}^{\text{out}} \in \mathbb{R}^{P \times P \times D_{out} \times D_{out}}$. The input embedding $\mathbf{v}_i$ is projected by

$$\mathbf{u}_{i,k} \in \mathbb{R}^{D_{out}} \leftarrow \mathbf{W}^{\text{in}}[\boldsymbol{\delta}_{i,k}]\mathbf{v}_i. \tag{5}$$

Then we use $\mathbf{u}_{i,k}$ in patch k as input to linear attention. Similarly, before aggregating the interim outputs, we also conduct a PAP, and Eq. (3) becomes

Algorithm 1 Spatially-Sparse Linear Attention (SSLA) Module Training

Require: Events $\mathcal{E}$, embeddings $\{\mathbf{v}_i\}_{i=1}^{L}$, coordinates $\{\mathbf{x}_i\}_{i=1}^{L}$, Patches $\mathcal{P}$, lookup table $T : \mathbf{x} \in \Omega \mapsto \{(k, \delta)\}$.
Ensure: Updated embeddings $\{\mathbf{o}_i\}_{i=1}^{L}$ in the same order as the input.
Initialize an empty embedding sequence $\mathcal{U}$ of length AL.
for each event e_i **in parallel do**
  Lookup $\mathbf{x}_i$: active patch indices $\{k \mid k \in \mathcal{K}_i\}$, relative positions $\{\boldsymbol{\delta}_{i,k} \mid k \in \mathcal{K}_i\}$;
  Position-aware projection: $\mathbf{u}_{i,k} \leftarrow \mathbf{W}^{\text{in}}[\boldsymbol{\delta}_{i,k}]\mathbf{v}_i$;
  Assign $\mathcal{U}[A(i-1) : Ai] \leftarrow \{\mathbf{u}_{i,k} \mid k \in \mathcal{K}_i\}$
end for
Scatter: stable-sort $\mathcal{U}$ based on k, and cache the permutation π;
Split $\mathcal{U}$ to patch-specific subsequences $\{\mathcal{U}_k \mid k \in \mathcal{P}\}$;
for each patch $k \in \mathcal{P}$ **in parallel do**
  $\mathcal{O}_k \leftarrow$ **LinearAttention**$(\mathcal{U}_k)$ using Eq. (1);
end for
Concat $\mathcal{O}_k$ into interim output sequence $\mathcal{O}$;
Gather: $\mathcal{O} \leftarrow \mathcal{O}[\pi^{-1}]$. We get $\mathcal{O}[A(i-1) : Ai] = \{\mathbf{o}_{i,k} \mid k \in \mathcal{K}_i\}$;
for each event e_i **in parallel do**
  Position-aware projection: $\mathbf{o}_{i,k} \leftarrow \mathbf{W}^{\text{out}}[\boldsymbol{\delta}_{i,k}]\mathbf{o}_{i,k}$;
end for
Aggregate: $\mathbf{o}_i = \sum \mathcal{O}[A(i-1) : Ai]$;
return $\{\mathbf{o}_i\}_{i=1}^{L}$

$$\mathbf{o}_i = \sum_{k \in \mathcal{K}_i} \mathbf{W}^{\text{out}}[\boldsymbol{\delta}_{i,k}]\mathbf{o}_{i,k}. \tag{6}$$

This projection encodes spatial priors with learnable parameters, expanding the capability of the model to capture spatial patterns and information.

Parallelizable Training with Scatter-Compute-Gather. While the MOS structure enables state-level sparsity, it breaks the single state form of linear attention, and thus poses challenges in efficient parallel training on GPUs. To address this, we derive a *scatter-compute-gather* algorithm, which allows both intra-patch (temporal) and inter-patch (spatial) parallelism to speed up training.

Scatter. Let $\mathbf{u}_{i,k}$ denote the projected embeddings of event e_i for the active patch $k \in \mathcal{K}_i$. We first reorganize the input event sequence $\mathcal{E}$ to K patch-specific subsequences, by constructing a subsequence for each patch k as

$$\mathcal{U}_k = \{\mathbf{u}_{i,k} \mid i \text{ such that } k \in \mathcal{K}_i\}, \quad \forall k \in \mathcal{P}, \tag{7}$$

where $\mathcal{U}_k$ is ordered by t_i. We implement this with a precomputed lookup table that maps each coordinate in Ω to the indices of the patches covering it, together with its relative position inside each patch. Using the table, we expand $\mathcal{E}$ to a projected sequence $\mathcal{U}$ of length AL, where each event contributes A consecutive projected embeddings, one for each of its activated patches. We apply

stable sorting on $\mathcal{U}$ based on the patch indices of each element, forming a reorganized sequence with embeddings in the same patch grouped together as the subsequence, while preserving the temporal order within each $\mathcal{U}_k$. The resulting permutation is cached and later reused in the gather step.

Compute. All patches share the same parameters but maintain independent states, so that the K subsequences can be computed in parallel, achieving inter-patch parallelism. We apply linear attention (Eq. (1)) to each subsequence $\mathcal{U}_k$

$$\mathcal{O}_k = \textbf{LinearAttention}(\mathcal{U}_k), \tag{8}$$

where $\mathcal{O}_k$ contains the interim outputs for events contained in patch k. In each patch, the linear attention is standard, which achieves intra-patch parallelism.

Gather. Finally, we restore the outputs to the original expanded event sequence order using the cached permutation and aggregate the interim outputs from all active patches of each event. Specifically, we first apply the PAP to each interim output $\mathbf{o}_{i,k}$, and then sum them over $k \in \mathcal{K}_i$ according to Eq. (6). This gather step is efficient because it only involves indexed reordering and reduction. The training procedure of the SSLA module is summarized in Algorithm 1.

3.4 SSLA-Det Architecture

We propose the SSLA-Det model for asynchronous event-based object detection. An overview of our neural network is shown in Fig. 3, which consists of an asynchronous backbone and a YOLOX detection head [10]. The backbone has 4 stages, each containing 2 SSLA module layers. In the SSLA layers, we use residual connections [14] and layer normalization [1] to stabilize training. In the first 3 stages, we use one sparse pooling and one temporal dropout layer introduced in [36], which compresses event sequence to reduce computation while preserving the high temporal resolution of events. All of the above layers are asynchronous, which makes the backbone fully asynchronous.

SSLA module is agnostic to the specific design of linear attention mechanism, allowing for the integration of any variant, including linear RNNs and SSMs. We use a real-valued Linear Recurrent Unit [27] implemented by Triton [42] in our model for hardware efficiency. The embedding dimension D_{out} has an expansion of $2\times$ in each stage.

For each step, the input to the model is a raw event, with polarity and time difference $\mathbf{v}_i = [p_i, \Delta t_i] \in \mathbb{R}^2$ as the embedding, and the backbone generates $\mathbf{o}_i$. We form a spatially fine-grained representation $\mathbf{R} \in \mathbb{R}^{H_{out} \times W_{out} \times D_{out}}$ from the asynchronous output, by updating $\mathbf{o}_i$ to $\mathbf{R}[\mathbf{x}_i]$. To achieve an end-to-end asynchronicity, we modify the YOLOX head by changing all the convolutions to 1×1. Each backbone output only updates the head predictions at position $\mathbf{x}_i$, making the YOLOX head also asynchronous. The whole SSLA-Det model is therefore end-to-end fully asynchronous, leading to minimal latency.

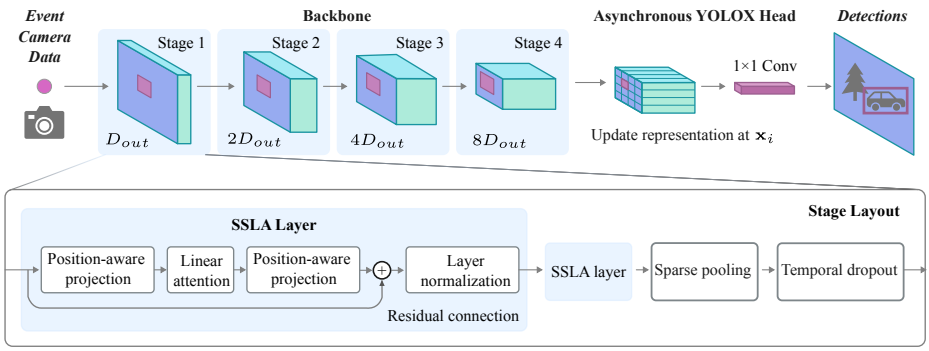


Fig. 3: Overview of the SSLA-Det model. Top: The *fully asynchronous* event-based object detector. Events are processed by a 4-stage asynchronous backbone, and each stage doubles the output embedding dimension. Each output embedding updates the representation of its position, and an asynchronous YOLOX head gives the detections. The red region marks the area sparsely activated by an event. **Bottom:** Stage layout. Each stage has 2 SSLA layers followed by sparse pooling and temporal dropout (only SSLA layers at stage 4). In one SSLA layer, event embeddings are processed by the SSLA module, including two position-aware projections and a patch-wise linear attention. A residual connection and a layer normalization are used for training stability.

4 Experiments

4.1 Experimental Setup

Datasets. Following previous work [11], we evaluate on the N-Caltech101 Detection [26] and the Gen1 Detection [5] datasets. N-Caltech101 consists of recordings captured by a DAVIS240 event camera with a resolution of 240×180 pixels, undergoing saccadic motion in front of a projector displaying Caltech101 images. Bounding box annotations were manually added in post-processing, with 101 classes. Gen1 is a more challenging, large-scale benchmark for automotive scenarios, recorded by an ATIS event camera with a resolution of 304×240 pixels. The dataset has two categories of 228,123 cars and 27,658 pedestrians. Following previous works [11, 34], we filter out bounding boxes with a diagonal below 30 pixels or width below 20 pixels in Gen1.

Training Details. All experiments were conducted with PyTorch 2.6.0 [30] on NVIDIA Ampere GPUs (A800/RTX 3090). We design four variants of our SSLA-Det model: small (SSLA-S), base (SSLA-B), medium (SSLA-M), and large (SSLA-L), by scaling the embedding dimension D_{out} of the first stage to 12, 16, 24 and 32. We use AdamW [24] optimizer. For Gen1, we train 40 epochs using a batch size of 32 and a base learning rate of 1×10^{-3} with a cosine decay. We use random flipping with probability 0.5 and random dropout of input events with the keep ratio sampled from $\mathcal{U}(0.8, 1.0)$. For N-Caltech101, we train 200 epochs with a batch size of 64. In addition to random dropout, we apply

Table 1: Object detection results on the Gen1 Detection dataset. Async. refers to asynchronous methods.

Method	Async.	mAP(↑)	AP_{50}(↑)	MFLOPS/ev(↓)
EVA+RVT-B [13]	✗	0.477	-	3.5 $\times 10^3$
GET [33]	✗	0.479	-	3.6 $\times 10^3$
SAST-CB [32]	✗	0.482	-	2.4 $\times 10^3$
SMamba [45]	✗	0.504	-	2.4 $\times 10^3$
ERGO-12 [49]	✗	0.504	-	50.8 $\times 10^3$
EventPillars [8]	✗	0.531	-	50.8 $\times 10^3$
NVS-S [22]	✓	-	0.086	7.80
AsyNet [25]	✓	-	0.145	205
AEGNN [37]	✓	-	0.163	5.26
FARSE-CNN [36]	✓	-	0.300	0.137
DAGr-N [11]	✓	0.263	-	1.36
DAGr-S [11]	✓	0.304	-	4.58
DAGr-M [11]	✓	0.318	-	9.94
DAGr-L [11]	✓	0.321	-	17.4
SSLA-S (Ours)	✓	0.334	0.629	**0.102**
SSLA-B (Ours)	✓	0.351	0.655	0.182
SSLA-M (Ours)	✓	0.370	0.670	0.408
SSLA-L (Ours)	✓	**0.375**	**0.675**	0.724

random cropping to 75% of the full resolution with probability 0.2 and random translation by up to 10% of the full resolution following the implementations of [11]. We also use exponential model averaging [17].

4.2 Results

Gen1 Automotive. We compare our SSLA-Det models with asynchronous event-based detection baselines [11, 22, 25, 36, 37], and use synchronous methods as reference [8, 13, 32, 33, 45, 49]. The performance is evaluated in accuracy with mean average precision (mAP) [23] and efficiency with average floating point operations (FLOPS) for every new event. We observed that some prior works report AP_{50} whereas others report mAP, and these values are sometimes presented together in the literature. To avoid potentially misleading comparisons, we separate them in Tab. 1.

Our SSLA-Det models consistently improve the accuracy-efficiency trade-off over existing asynchronous baselines. Notably, compared to the strongest prior asynchronous baseline DAGr-L [11], our smallest model SSLA-S achieves higher mAP (0.334 vs. 0.321) while reducing the computational cost by about 171× (0.102 vs. 17.4 MFLOPS/ev). Our largest model, SSLA-L, achieves an mAP of 0.375, setting a new SOTA for asynchronous detection on Gen1, with $> 20\times$ reduction of FLOPS to previous best model DAGr-L (0.724 M/ev vs. 17.4 M/ev).

Fig. 4 visualizes the detection results on Gen1. Fig. 4 (a)-(d) and Fig. 4 (e)-(h) show the detected cars and pedestrians, respectively. Fig. 4 (i)-(l) provide

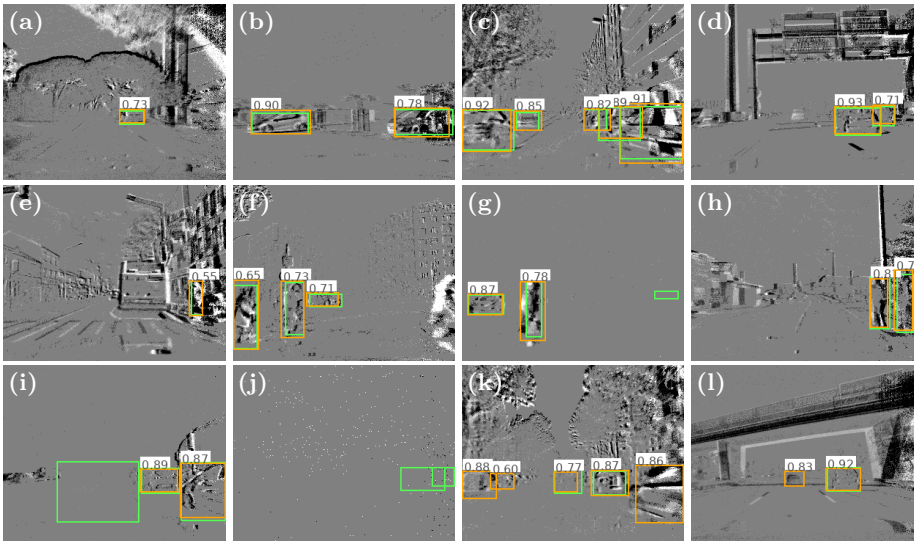


Fig. 4: Visualization of the detection results on the Gen1 dataset. Green boxes denote ground truth and orange boxes denote predictions with confidence scores. Predicted boxes with confidence scores below 0.5 are removed. (a)-(d): Cars. (e)-(h): Pedestrians. (i)-(l): Failure cases.

qualitative understandings of the typical failure cases. Specifically, Fig. 4 (i) and (j) show the false negatives, mainly caused by the lack of relative motion between the event camera and the target, which leads to missing event data. Fig. 4 (k) and (l) show the false positives caused by missing annotations.

N-Caltech101. Tab. 2 presents the performance of our models on the N-Caltech101 dataset. Consistent with the Gen1 results, our models achieve a superior accuracy-efficiency trade-off among asynchronous methods. In particular, SSLA-L reaches an AP_{50} of 0.743 with a computational cost of only 0.926 MFLOPS/ev. Compared with the previous best asynchronous baseline DAGr-L, SSLA-L improves AP_{50} by 1.1 points (0.743 vs. 0.732) while using 20× fewer MFLOPS per event (0.926 vs. 18.9).

4.3 Timing Experiments

Training Efficiency. We show the training efficiency benefit of linear attention with sequential parallelism. We replace linear attention with a long short-term memory (LSTM) baseline [16] with the same hidden dimension as SSLA-S. Tab. 3 compares SSLA with an LSTM baseline from the official PyTorch implementation under the same training setup on Gen1. We compare the training time per epoch, which is measured on 4 NVIDIA A800 GPUs. SSLA-S reduces the epoch time from 1.05 to 0.25 hours (4.2×), but at the cost of a drop in mAP (from

Table 2: Object detection results on the N-Caltech101 Detection dataset. Async. refers to asynchronous methods.

Method	Async.	mAP(↑)	AP_{50}(↑)	MFLOPS/ev(↓)
NVS-S [22]	✓	-	0.346	7.80
AsyNet [25]	✓	-	0.643	200
AEGNN [37]	✓	-	0.595	7.41
EHGCN [2]	✓	-	0.694	1.06
DAGr-N [11]	✓	-	0.629	2.28
DAGr-S [11]	✓	-	0.702	6.85
DAGr-M [11]	✓	-	0.727	12.2
DAGr-L [11]	✓	-	0.732	18.9
SSLA-S (Ours)	✓	0.444	0.681	**0.131**
SSLA-B (Ours)	✓	0.483	0.720	0.233
SSLA-M (Ours)	✓	0.495	0.724	0.522
SSLA-L (Ours)	✓	**0.515**	**0.743**	0.926

0.353 to 0.334), mainly caused by a lower FLOPS. At a similar FLOPS level, SSLA-B achieves a comparable mAP to LSTM (0.351 vs. 0.353) and higher AP_{50} (0.655 vs. 0.631), while reducing train time from 1.05 to 0.28 hours (3.8×).

Table 3: Comparison of training efficiency between SSLA and an LSTM baseline on Gen1. Training time per epoch is measured on 4 NVIDIA A800 GPUs.

Method	Time/epoch (h)(↓)	mAP(↑)	AP_{50}(↑)	Params (M)	MFLOPS/ev(↓)
SSLA-S	**0.25**	0.334	0.629	0.508	**0.102**
SSLA-B	0.28	**0.351**	**0.655**	0.902	0.182
LSTM	1.05	**0.353**	0.631	0.606	0.176

Inference Latency. We measure the latency of SSLA-Det as the time required to process one newly arrived event in a recurrent, event-by-event setting. We implement a recurrent C++ version of SSLA-Det and benchmark the latency on a single core of an AMD Ryzen 9 9950X3D CPU. As shown in Tab. 4, our models achieve a low latency of less than 10 μs, which is lower than the sensor transmission latency of approximately 200 μs [11]. Interestingly, a smaller model does not yield lower latency in our setting (*e.g.* on Gen1, 3.43 μs for SSLA-S and 2.44 μs for SSLA-B), because the actual runtime also depends on hardware factors such as vectorization efficiency and memory access patterns. The SSLA module has a constant per-event inference FLOPS of $\mathcal{O}(P^2 D_{out}^2)$, making the latency independent of resolution. While CPU does not fully translate our FLOPS efficiency into latency gains [11], further runtime latency reduction could be achieved on specific hardware, such as FPGAs [18] or neuromorphic accelerators [48].

Table 4: Inference latency. Latency refers to the time used for the recurrent model to process a new event.

Dataset	Resolution	Latency (μs)			
		SSLA-S	SSLA-B	SSLA-M	SSLA-L
Gen1	304 × 240	3.43	2.44	6.02	7.20
N-Caltech101	240 × 180	3.60	2.61	6.50	8.01

4.4 Ablation Study

Efficiency Attribution. To isolate the sources of efficiency in SSLA-Det, we compare SSLA-S with three variants: (i) removing temporal dropout (TD), (ii) further replacing the SSLA module with a dense-activation counterpart that retains the MOS decomposition and PAP but activates all patches per event, and (iii) removing sparse pooling (SP). As shown in Tab. 5, the SSLA module contributes the dominant computational cost reduction (380 ×) at no accuracy cost, while TD provides an additional 10 × reduction as an accuracy-efficiency trade-off. SP does not change the per-event FLOPS since it only downscales the coordinates of events without reducing the event count, but is essential for detection.

Table 5: Efficiency Attribution. We isolate the contribution of SSLA-Det components on the validation set of Gen1. TD refers to temporal dropout, SP refers to sparse pooling, and we remove SSLA by changing it to its dense counterpart and keeping the MOS and PAP designs.

Configuration	SSLA-S	w/o TD	w/o SSLA & TD	w/o SP
MFLOPS/ev (↓)	0.102	1.02	388	0.102
mAP (↑)	0.335	0.370	0.370	0.014

Effect of Spatial Sparsity. We replace the SSLA module with a standard linear attention (LRU). We use D_{out} of the first stage 12 and 36, to keep same D_{out} and similar FLOPS as SSLA-S. Tab. 6 shows that using a standard linear attention fails in both cases, which is considered mainly due to the lack of fine-grained state. In particular, SSLA-S maintains a state 380 × larger than LA ($D_{out} = 36$) with similar FLOPS (0.102 M/ev vs. 0.093 M/ev). This demonstrates the importance of state-level sparsity.

Effect of Position-Aware Projection. To ablate PAP, we replace it with a position-irrelevant learnable linear projection. The results are summarized in Tab. 7. Removing either input or output PAP causes a significant accuracy drop,

Table 6: Effect of Spatial Sparsity. We use linear attention (LA) with same embedding dimension (first stage $D_{out} = 12$) and similar FLOPS (first stage $D_{out} = 36$) compared to SSLA-S. We report accuracy on the validation set of Gen1. State refers to the size of hidden state in the last layer, which reflects the capability to model fine-grained spatial representations.

Model	mAP(↑)	AP_{50}(↑)	AP_{75}(↑)	MFLOPS/ev(↓)	Params (M)	State (K)
SSLA-S	**0.335**	**0.610**	**0.322**	0.102	0.508	106.9
LA ($D_{out} = 12$)	0.001	0.004	0.000	**0.011**	0.130	0.094
LA ($D_{out} = 36$)	0.001	0.003	0.000	0.093	1.20	0.281

and removing both results in catastrophic failure, with the mAP collapsing to only 0.014, which highlights its importance for encoding spatial priors in the SSLA module.

Table 7: Effect of Position-Aware Projection. Input and Output refer to the PAP with $\mathbf{W}^{\text{in}}$ and $\mathbf{W}^{\text{out}}$, respectively. We report accuracy on the validation set of Gen1.

Position-Aware Projection		mAP(↑)	AP_{50}(↑)	AP_{75}(↑)	MFLOPS/ev(↓)
Input	Output				
✓	✓	**0.335**	**0.610**	**0.322**	0.102
✓		0.306	0.582	0.280	0.085
	✓	0.224	0.473	0.184	0.089
		0.014	0.048	0.006	**0.072**

Effect of Patch Size. P controls the receptive field of event interaction. As shown in Tab. 8, a smaller patch size ($P = 2$) reduces FLOPS (0.047 M/ev) but leads to a mAP drop (0.200). Conversely, $P = 4$ boosts the mAP to 0.371 but increases the FLOPS to 0.179 M/ev, showing an accuracy-efficiency trade-off. Besides, for training, increasing P results in higher GPU memory consumption and longer training time. Therefore, we select $P = 3$ as our default configuration as it yields a reasonable trade-off between efficiency and accuracy.

Table 8: Effect of Patch Size. We report accuracy on the validation set of Gen1.

Patch Size	mAP	AP_{50}	AP_{75}	MFLOPS/ev
$P = 2$	0.200	0.440	0.150	**0.047**
$P = 3$	0.335	0.610	0.322	0.102
$P = 4$	**0.371**	**0.656**	**0.364**	0.179

5 Limitation and Discussion

This work focuses on event-based low-latency object detection. While hybrid event-image models have become a recent research trend [11, 21], they also introduce additional challenges, including event-image alignment, extra sensor requirements, and high system complexity. Besides, in principle, our method is also compatible with the hybrid framework, since image features from dense models can be injected into the intermediate layers of our model. Exploring this event-image fusion in SSLA is an interesting direction for our future work.

Although SSLA-Det achieves SOTA performance among asynchronous event-based object detection methods, a gap remains compared with synchronous methods. For example, as shown in Tab. 1 on Gen1, SSLA-L has an mAP of 0.375, whereas synchronous SOTA methods exceed 0.5 mAP. This gap is expected, as asynchronous and synchronous methods target fundamentally different objectives along the accuracy-efficiency trade-off and are not directly comparable. Synchronous methods accumulate events into image-like representations and perform dense image-level inference, which allows for more information aggregation, the use of image-based neural network architectures and pretrained weights [13,49], and models with larger parameter count [8,49], but at the cost of larger computational cost (Tab. 1) and millisecond-level latency. Asynchronous models, in contrast, aim to realize the low-latency advantage of event cameras at the neural network level, giving predictions event-by-event at minimal latency. This structurally constrains parameter count, information aggregation, and architectural choices, naturally limiting accuracy. Therefore, the remaining accuracy gap should be understood as part of the accuracy-latency trade-off in low-latency event-based perception, rather than a methodological shortcoming. Further improving this trade-off while preserving μs-level per-event latency remains an important direction for future work.

6 Conclusion

In this paper, we propose SSLA, a novel linear attention module with spatial sparsity and efficient parallel training capability for event sequence modeling. We develop SSLA-Det, the first end-to-end asynchronous linear attention-based model for event-based object detection. Experimental results on Gen1 and N-Caltech101 show that SSLA-Det achieves SOTA asynchronous accuracy with significantly lower FLOPS than previous asynchronous baselines. We believe that SSLA provides a promising direction for low-latency, high-performance event-based perception.

Acknowledgements

This work was supported by the State Key Laboratory of Precision Measurement Technology and Instruments (2025PMTI03), and STI 2030-Major Projects (2021ZD0200300).

References

1. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
2. Chen, H., Luo, L., Mo, M., Wu, Z., Xiao, G., Gan, J., Leng, J., Gao, X.: Ehgcn: Hierarchical euclidean-hyperbolic fusion via motion-aware gcn for hybrid event stream perception. arXiv preprint arXiv:2504.16616 (2025)
3. Dalgaty, T., Mesquida, T., Joubert, D., Sironi, A., Vivet, P., Posch, C.: 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. pp. 3953–3962 (2023)
4. Dampfhoffer, M., Mesquida, T., Joubert, D., Dalgaty, T., Vivet, P., Posch, C.: Graph neural network combining event stream and periodic aggregation for low-latency event-based vision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6909–6918 (2025)
5. De Tournemire, P., Nitti, D., Perot, E., Migliore, D., Sironi, A.: A large scale event-based detection dataset for automotive. arXiv preprint arXiv:2001.08499 (2020)
6. Du, J., Sun, W., Lan, D., Hu, J., Zhang, T., Cheng, Y.: Mom: Linear sequence modeling with mixture-of-memories. In: The Fourteenth International Conference on Learning Representations (2026)
7. Falanga, D., Kleber, K., Scaramuzza, D.: Dynamic obstacle avoidance for quadrotors with event cameras. Science Robotics **5**(40), eaaz9712 (2020)
8. Fan, R., Hao, W., Guan, J., Rui, L., Gu, L., Wu, T., Zeng, F., Zhu, Z.: Eventpillars: Pillar-based efficient representations for event data. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 2861–2869 (2025)
9. 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)
10. Ge, Z., Liu, S., Wang, F., Li, Z., Sun, J.: Yolox: Exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430 (2021)
11. Gehrig, D., Scaramuzza, D.: Low-latency automotive vision with event cameras. Nature **629**(8014), 1034–1040 (2024)
12. Gu, A., Dao, T.: Mamba: Linear-time sequence modeling with selective state spaces. In: First conference on language modeling (2024)
13. Hao, H., Zubic, N., He, W., Sui, Z., Scaramuzza, D., Wang, W.: Maximizing asynchronicity in event-based neural networks. In: The Fourteenth International Conference on Learning Representations (2026)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
15. He, W., Zhu, J., Feng, Y., Liang, F., You, K., Chai, H., Sui, Z., Hao, H., Li, G., Zhao, J., et al.: Neuromorphic-enabled video-activated cell sorting. Nature communications **15**(1), 10792 (2024)
16. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
17. Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., Wilson, A.G.: Averaging weights leads to wider optima and better generalization. arXiv preprint arXiv:1803.05407 (2018)

18. Jeziorek, K., Wzorek, P., Blachut, K., Nakano, H., Dampfhoffer, M., Mesquida, T., Nishi, H., Dalgaty, T., Kryjak, T.: Hardware-accelerated graph neural networks: an alternative approach for neuromorphic event-based audio classification and keyword spotting on soc fpga. arXiv preprint arXiv:2602.16442 (2026)
19. 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 (2023)
20. Katharopoulos, A., Vyas, A., Pappas, N., Fleuret, F.: Transformers are rnns: Fast autoregressive transformers with linear attention. In: International conference on machine learning. pp. 5156–5165. PMLR (2020)
21. Li, D., Li, J., Liu, X., Fan, X., Tian, Y.: Asynchronous collaborative graph representation for frames and events. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 1655–1666 (2025)
22. 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)
23. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)
24. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2019)
25. 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)
26. Orchard, G., Jayawant, A., Cohen, G.K., Thakor, N.: Converting static image datasets to spiking neuromorphic datasets using saccades. Frontiers in neuroscience **9**, 437 (2015)
27. Orvieto, A., Smith, S.L., Gu, A., Fernando, A., Gulcehre, C., Pascanu, R., De, S.: Resurrecting recurrent neural networks for long sequences. In: International conference on machine learning. pp. 26670–26698. PMLR (2023)
28. Pan, Y., An, Y., Li, Z., Chou, Y., Zhu, R.J., Wang, X., Wang, M., Wang, J., Li, G.: Scaling linear attention with sparse state expansion. In: The Fourteenth International Conference on Learning Representations (2026)
29. Paredes-Vallés, F., Hagenaars, J.J., Dupeyroux, J., Stroobants, S., Xu, Y., de Croon, G.C.: Fully neuromorphic vision and control for autonomous drone flight. Science Robotics **9**(90), eadi0591 (2024)
30. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
31. Peng, B., Alcaide, E., Anthony, Q., Albalak, A., Arcadinho, S., Biderman, S., Cao, H., Cheng, X., Chung, M., Derczynski, L., et al.: Rwkv: Reinventing rnns for the transformer era. In: Findings of the association for computational linguistics: EMNLP 2023. pp. 14048–14077 (2023)
32. Peng, Y., Li, H., Zhang, Y., Sun, X., Wu, F.: Scene adaptive sparse transformer for event-based object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16794–16804 (2024)
33. Peng, Y., Zhang, Y., Xiong, Z., Sun, X., Wu, F.: Get: Group event transformer for event-based vision. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6038–6048 (2023)

34. 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)
35. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
36. Santambrogio, R., Cannici, M., Matteucci, M.: Farse-cnn: Fully asynchronous, recurrent and sparse event-based cnn. In: European conference on computer vision. pp. 1–18. Springer (2024)
37. 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)
38. Schöne, M., Sushma, N.M., Zhuge, J., Mayr, C., Subramoney, A., Kappel, D.: Scalable event-by-event processing of neuromorphic sensory signals with deep state-space models. In: 2024 International Conference on Neuromorphic Systems (ICONS). pp. 124–131. IEEE (2024)
39. 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)
40. Sekikawa, Y., Nagata, J., Araki, I., Girbau, A.: Col2a: Convolution-free local linear attention for spatiotemporal event processing. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 4869–4880 (2026)
41. Soydan, T., Zubić, N., Messikommer, N., Mishra, S., Scaramuzza, D.: S7: Selective and simplified state space layers for sequence modeling. arXiv preprint arXiv:2410.03464 (2024)
42. Tillet, P., Kung, H.T., Cox, D.: Triton: an intermediate language and compiler for tiled neural network computations. In: Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages. pp. 10–19 (2019)
43. Turrero, C.M., Bouvier, M., Breitenstein, M., Zanuttigh, P., Parret, V.: ALERT-transformer: Bridging asynchronous and synchronous machine learning for real-time event-based spatio-temporal data. In: Forty-first International Conference on Machine Learning (2024)
44. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
45. Yang, N., Wang, Y., Liu, Z., Li, M., An, Y., Zhao, X.: Smamba: Sparse mamba for event-based object detection. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 9229–9237 (2025)
46. Yang, S., Wang, B., Shen, Y., Panda, R., Kim, Y.: Gated linear attention transformers with hardware-efficient training. In: International Conference on Machine Learning. pp. 56501–56523. PMLR (2024)
47. Yang, S., Wang, B., Zhang, Y., Shen, Y., Kim, Y.: Parallelizing linear transformers with the delta rule over sequence length. Advances in neural information processing systems **37**, 115491–115522 (2024)
48. Zhang, X., Hu, M., Lu, S., Kim, S., Lee, E.Y.J., Liu, Y., Lu, W.D.: Compute-in-memory implementation of state space models for event sequence processing. Nature Communications **17**(1), 1513 (2026)
49. Zubić, N., Gehrig, D., Gehrig, M., Scaramuzza, D.: From chaos comes order: Ordering event representations for object recognition and detection. In: Proceedings

of the IEEE/CVF International Conference on Computer Vision. pp. 12846–12856 (2023)
50. Zubić, N., Gehrig, M., Scaramuzza, D.: State space models for event cameras. In: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5819–5828. IEEE (2024)