Scene Adaptive Sparse Transformer for Event-based Object Detection

Yansong Peng[1,*] Hebei Li[1,*] Yueyi Zhang[1,†] Xiaoyan Sun[1,2] Feng Wu[1,2]
[1]University of Science and Technology of China
[2]Institute of Artificial Intelligence, Hefei Comprehensive National Science Center
{pengyansong, lihebei}@mail.ustc.edu.cn, {zhyuey, sunxiaoyan, fengwu}@ustc.edu.cn

Abstract

While recent Transformer-based approaches have shown impressive performances on event-based object detection tasks, their high computational costs still diminish the low power consumption advantage of event cameras. Image-based works attempt to reduce these costs by introducing sparse Transformers. However, they display inadequate sparsity and adaptability when applied to event-based object detection, since these approaches cannot balance the fine granularity of token-level sparsification and the efficiency of window-based Transformers, leading to reduced performance and efficiency. Furthermore, they lack scene-specific sparsity optimization, resulting in information loss and a lower recall rate. To overcome these limitations, we propose the Scene Adaptive Sparse Transformer (SAST). SAST enables window-token co-sparsification, significantly enhancing fault tolerance and reducing computational overhead. Leveraging the innovative scoring and selection modules, along with the Masked Sparse Window Self-Attention, SAST showcases remarkable scene-aware adaptability: It focuses only on important objects and dynamically optimizes sparsity level according to scene complexity, maintaining a remarkable balance between performance and computational cost. The evaluation results show that SAST outperforms all other dense and sparse networks in both performance and efficiency on two large-scale event-based object detection datasets (1Mpx and Gen1). Code: https://github.com/Peterande/SAST.

1. Introduction

Event cameras asynchronously capture the illumination changes of each pixel in a bionic way and possess several advantages, such as high temporal resolution (>10K fps) and a wide dynamic range (>120 dB) [12]. Unlike frame cameras that capture the whole scene at a fixed rate, event cameras exclusively record sparse event streams when objects are in motion, resulting in no events generated in static scenes. This distinctive trait enables event cameras to operate with low power consumption (<10 mW) [12].

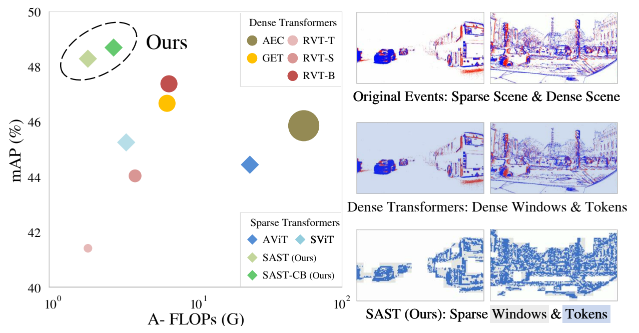

Figure 1. **Detection performance vs. computational cost** on 1Mpx, with marker size indicating model size. SAST exhibits superiority by employing window-token co-sparsification and scene-specific sparsity optimization, maintaining low computation while delivering high performance across varying scenes.

Building on these unique advantages of event cameras, event-based object detection excels in challenging motion and lighting conditions, and offers energy-efficient solutions in power-constrained environments [20, 28, 31, 41, 43, 63]. However, as the raw events are asynchronous, traditional Image-based networks can not be directly applied to them. To overcome this problem, prior works often convert events into Image-like representations, such as event voxel [67], event histogram [39], and time surface [26, 51] first. Features are extracted from these representations using different neural network architectures, including CNN-based [10, 13, 14, 33], SNN-based [6, 9, 27, 34, 59], and GNN-based networks [30, 49]. Recently, many innovative works have demonstrated that Vision Transformers can achieve superior performance on event-based object detection tasks [15, 16, 42]. However, the quadratic computational complexity of self-attention in Transformers hinders model scalability [52], which challenges the balance between performance and efficiency for event-based object detection. The high power consumption of such dense operation also di-

*Joint first author †Corresponding author minishes the low power consumption advantage of event cameras. Additionally, we observe that due to the high sparsity of events, a majority of the computational cost of the Transformer is unnecessary. Especially for high-resolution event streams, such as those in the 1Mpx [43] dataset, while higher resolution brings more detail, event sparsity can fluctuate by several hundred times. During self-attention computations, the blank regions without any events are also encoded as tokens and participate in calculations. Our goal is to reduce this unnecessary computational overhead.

We revisit the mainstream Image-based sparse Transformers, which are also designed to reduce the computational overhead of self-attention. Most of them achieve token-level sparsification based on global self-attention [2, 3, 22, 24, 35, 40, 45, 61]. Some studies among them achieve adaptive sparsification [35, 61], but they are unable to choose the optimal sparsity for individual samples. What's more, despite computing much fewer tokens than the original ViT [8], they still require significantly more computational resources than window-based Transformers [36, 37, 54, 65] that leverage compute-efficient window self-attention. Consequently, the computational cost remains a barrier for high-resolution object detection tasks. SparseViT (SViT) [4] proposes a sparse window-based Transformer architecture. However, it relies on a manually selected window pruning ratio, which may discard important windows containing objects. This results in poor robustness in certain scenarios, particularly when many objects smaller than the window scale are present.

In this work, we introduce Scene Adaptive Sparse Transformer (SAST) for event-based object detection. SAST achieves window-token co-sparsification in a window-based Transformer architecture. As shown in Fig. 1, SAST greatly reduces computational overhead (A-FLOPs: Attention-related FLOPs, exclusive of the computations incurred by convolutional layers.), while simultaneously enhancing performance. Moreover, it leverages the scoring and selection modules, realizing scene-specific sparsity optimization which can adaptively choose the optimal sparsification strategy based on the complexity of different scenes. We also propose the Masked Sparse Window Self-Attention (MS-WSA), which efficiently performs self-attention on selected tokens with unequal window sizes and isolates all context leakage to achieve optimal performance. Our main contributions are summarized in the following.

- We develop a highly efficient and powerful SAST for event-based object detection, which maintains a remarkable balance between performance and efficiency.
- We propose innovative scoring and selection modules, which assess the importance of each window and token and perform co-sparsification on them.
- We devise the MS-WSA, which efficiently performs window self-attention on selected tokens with unequal window sizes and avoids context leakage.
- Experimental results on 1Mpx and Gen1 datasets demonstrate the superiority of SAST, surpassing all dense and sparse networks.

2. Related Work

Vision Transformers. The Vision Transformer (ViT) [8] stands as a seminal work providing a vision backbone that applies self-attention to images. Based on that, many ViT variants have emerged to enhance performance and efficiency. For example, linear Transformers [5, 44, 56, 60] are explored for finding the approximation of self-attention. Other works [17, 29, 36, 54, 62] introduce localized self-attention or hierarchical Transformer architectures.

Sparse Transformers. Sparse Transformers are proposed to improve the efficiency of ViTs by selectively computing self-attention on partial tokens. Most of them achieve token-level sparsification based on traditional ViT [2, 3, 22, 24, 35, 40, 45, 61]. However, these sparsified networks still require significantly more computational resources than window-based Transformers [36, 54]. SViT [4] proposes a window-level sparse Transformer based on Swin Transformer [36]. However, its manually selected window pruning ratio leads to the unwanted discarding of important windows. If the scene is dense or the objects are smaller than the window scale, performance will suffer severely.

Event-Based Transformers. Transformer networks have achieved high performance across various event-based tasks, including but not limited to classification [48], object detection [15, 42], and semantic segmentation [16]. However, due to its high computational complexity, many early attempts only incorporate self-attention operations on feature maps derived from CNN backbones [23, 53, 58, 66]. Recently, alternative approaches [15, 42, 48, 57, 64] using efficient window Transformers aim to extract features from event representations directly [26, 39, 51, 67]. However, a substantial computational burden of self-attention still exists for high-resolution tasks [15, 16], which limits the model's scalability and performance [52].

Event-Based Object Detection The early efforts in event-based object detection involve converting event streams into images and feeding them into existing Image-based detectors [1, 25]. Significant strides were made with the advent of benchmark large-scale datasets 1Mpx [43] and Gen1 [7]. They paved the way for the development of innovative networks such as RED [43] and ASTMNet [28], which introduce memory mechanisms to fully exploit the spatiotemporal information within event data. Some approaches based on SNNs and GNNs were also proposed [6, 49], but their performance still lags behind CNN-based methods. Recently, remarkable performance has been achieved by RVT [15], HMNet [16], and GET [42] using Transformer networks. However, these networks still involve significant re-

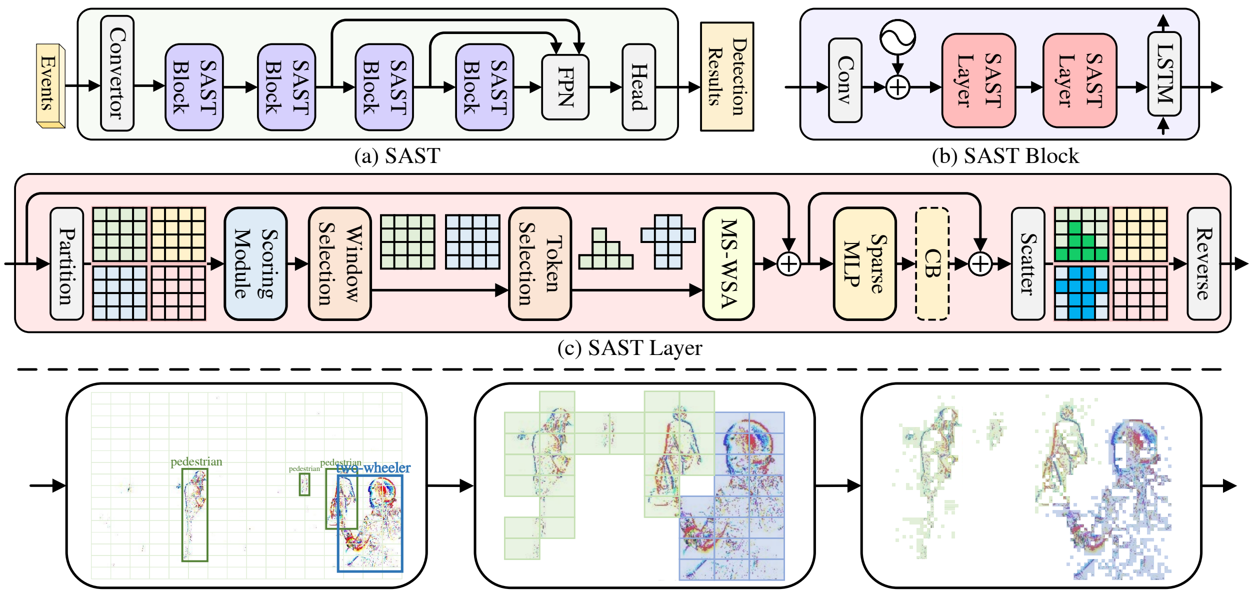

Figure 2. **(a) The hierarchical architecture of SAST.** Four SAST blocks extract multi-scale features from sparse tokens transformed from events. **(b) The architecture of SAST block**, which contains two successive SAST layers. **(c) The architecture of SAST layer.** In an SAST layer, tokens are partitioned into windows and scored by the scoring module first. The selection module selects important windows and tokens. Then, the selected tokens within selected windows are sequentially processed through MS-WSA, a sparse MLP layer, and an optional CB operation. Finally, processed tokens are scattered back and reversed from windows. Norm layers are omitted for simplification.

dundancy in self-attention computations of sparse events.

3. Method

3.1. Main Architecture

The overall architecture of SAST is shown in Fig. 2(a). The asynchronous events are initially converted into event voxel representations [67]. These event voxels are fed into four hierarchical SAST blocks illustrated in Fig. 2(b) to extract multi-scale features. Within the first block, a convolutional layer, as in ViT [8], transforms the event voxel into tokens. Subsequently, the sinusoidal positional encoding is added to the tokens. Two successive SAST layers extract spatial features from these tokens. At the end of the SAST block, an LSTM layer propagates temporal information, with its output sent to subsequent layers. In the following blocks, tokens undergo a convolutional layer first to reduce the spatial resolution, the remaining process being the same as in the first block. The extracted features from the second, third, and fourth blocks are relayed to the Feature Pyramid Network (FPN). The FPN then forwards the processed features to the detection head, yielding the final detection results.

3.2. SAST Layer

As shown in Fig. 2(c), in an SAST layer, all the tokens are partitioned into windows first. After that, a scoring module scores the partitioned tokens, determining their importance. The selection module selects important windows and tokens inside windows sequentially based on these scores. The token selection results are shared between SAST layers within the same block. Next, the Masked Sparse Window Self-Attention (MS-WSA) is performed on selected tokens within selected windows. The attention-enhanced tokens are sequentially processed by a sparse MLP layer and an optional context broadcasting (CB) operation [19]. The sparse MLP layer indicates that sparse connections are established exclusively on selected tokens, thereby benefiting from sparsification and reducing computational load. Finally, the processed tokens are scattered back and reversed from windows to their original shape.

The SAST layer achieves sparsification on both windows and tokens, ensuring efficiency and preventing important windows containing objects from being discarded. Its scene-aware adaptability also guides the network to focus more on important areas, resulting in better performance. What's more, the computational cost of SAST is fully dynamic across different scenes, while maintaining an overall low level. The SAST layer is compatible with different Transformer architectures, which means it can be plugged as a universal layer in any hierarchical Transformer. We then detail the three main components of the SAST layer: the scoring module, the selection module, and MS-WSA.

3.3. Scoring Module

As illustrated in Fig. 3(a), the scoring module is designed to score each token, determining its importance. Unlike other

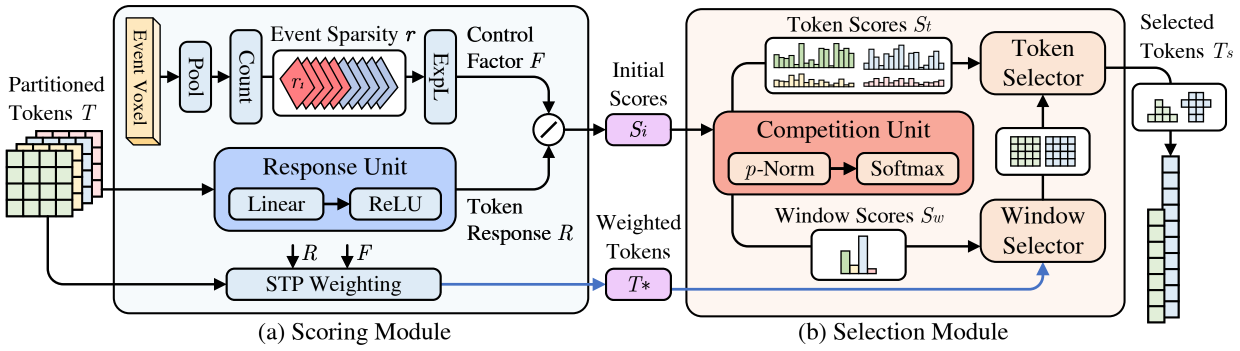

Figure 3. **The architecture of scoring and selection modules.** (a) The scoring module scores each window and token. The scoring process is regulated based on event sparsity, with windows and tokens competing to limit selections. (b) The selection module uses two filters to select important windows and tokens sequentially based on their scores.

sparse Transformers obtaining scores directly from the token value or attention map, SAST has a controllable and learned scoring module for reasonable scoring.

Initially, the partitioned tokens T are sent to a response unit to obtain their token responses R. The response unit consists of a linear layer followed by a ReLU layer. On a parallel branch, due to multi-scale design, the original event voxel is downsampled by a pooling layer first to match the receptive field of tokens. Event sparsity $\mathbf{r}$ is obtained by calculating and concatenating B non-zero ratios $(r_1, r_2, ..., r_B)$ of different voxel bins. Each ratio reflects the sparsity of a specific subset of events with unique time ranges and polarities. Then, the event sparsity $\mathbf{r}$ is projected to the same dimension as the tokens through the Exponential Linear layer (ExpL), resulting in the control factor F. Based on the definition of the token response R and the control factor F, the initial score S_i is defined as:

$$S_i = a \cdot \frac{R}{F} = a \cdot \frac{\text{ReLU}(W_R \cdot T + b_R)}{\exp(W_F) \cdot \mathbf{r}}, \quad (1)$$

where W_R and b_R represent the weight and bias of the linear layer in the response unit, while W_F represents the weight of the ExpL. The function $\exp(\cdot)$ denotes the exponential operation, which is involved to ensure that the control factor F is positive. a is a hyper-parameter controlling the absolute sparsity level of SAST. According to our observation, tokens corresponding to more important objects tend to have higher initial scores.

To make the response unit and the ExpL learnable, we apply Spatial-Temporal-Polar (STP) weighting to the partitioned tokens T. A Sigmoid layer transforms the token response R and the control factor F into spatial weight W_s and temporal-polar weight W_{tp}. The weighted tokens $T*$ are produced through the product of these weights, as described by the equation:

$$T* = W_{tp} \cdot W_s \cdot T = \text{Sigmoid}(R) \cdot \text{Sigmoid}(F) \cdot T. \quad (2)$$

The STP weighting process allows the model to emphasize tokens based on their spatial and temporal-polar context, aligning the network's sparse processing with the most salient features across both domains.

3.4. Selection Module

As illustrated in Fig. 3(b), the selection module is designed to select important tokens and windows. However, the initial scores derived from normalized tokens show insufficient contrast for effective differentiation. Therefore, the competition unit is designed to intensify the competition among the S_i. It first calculates the p-norm of the S_i, obtaining the normalized score for each token and window, respectively. The choice of p determines the exponential relationship between the normalized score and event sparsity. Softmax operation is further applied to amplify the disparities between normalized scores, making the dominance of some values more pronounced. The intensified scores are calculated using the equations:

$$\begin{aligned} S_t &= \text{Softmax}(||S_i||_p^{\text{c}}) \\ S_w &= \text{Softmax}(||S_i||_p^{\text{c, w}}/N_t) \end{aligned} \quad (3)$$

where S_t and S_w are the token scores and window scores, respectively. $||.||_p^{\text{c}}$ denotes computing p-norm along the channel dimension, and $||.||_p^{\text{c,w}}$ denotes computing p-norm along both the channel and window dimensions. The division by N_t serves to scale the normalized window scores by the number of tokens in a window.

If the scene is sparse, fewer scores dominate, making it easier to select less important windows and tokens. Conversely, in dense scenes, the score distribution is relatively uniform, preserving more windows and tokens to prevent the loss of important object information. Then, the selection module specifies two thresholds μ_t and μ_w for the selection process, defined as:

$$\mu_t = \frac{b}{N_t}, \quad \mu_w = \frac{b}{N_w}, \quad (4)$$

where N_t is the number of tokens in a window, and N_w is the number of windows. b is a parameter in the range [0.9,1]

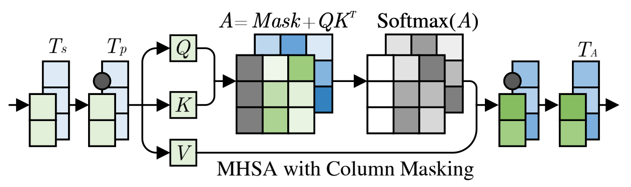

Figure 4. **Masked Sparse Window Self-Attention (MS-WSA).** An illustration of MS-WSA with window size equal to 2×2. Three and two tokens are selected from two windows. Due to the column masking, the padded token marked as a grey dot does not contribute to the attention map.

that controls the strictness of selection. Therefore, μ_t and μ_w correspond to values slightly smaller than the mean of token and window scores. Scores exceeding the threshold reflect their significance, ensuring that the selected proportion aligns with scene complexity.

In Fig. 3(b), weighted tokens $T*$ undergo an initial selection by a window selector that retains tokens from windows with scores S_w exceeding μ_w. Then, a token selector further refines this selection, keeping tokens with scores S_t surpass μ_t. The final selected tokens are represented as T_s.

3.5. Masked Sparse Window Self-Attention

Traditional WSA is designed for equal-sized windows, conducting parallel matrix multiplication on them. As a result, it cannot be applied to selected tokens with unequal window sizes. Therefore, we design MS-WSA to efficiently perform self-attention on selected tokens, integrating correlations among them and isolating context leakages.

As shown in Fig. 4, MS-WSA first pads the selected tokens T_s within each window to the same length. The padding operation involves selecting a minimal number of tokens from the filtered-out tokens. The padded tokens are represented as T_p. Afterward, we apply multi-head self-attention (MHSA) [55] in parallel to selected windows containing padded tokens. However, the tokens used for padding, originally unselected, also contribute to the attention map calculation. Moreover, the padding number is determined by the largest window in the batch. These uncertainties lead to contextual leakage both among tokens and across different batches. To isolate these context leakages, MS-WSA includes an additional masking operation when calculating the attention map. The overall MS-WSA process can be described as:

$$\begin{aligned} &T_p = \text{Pad}(T_s) \\ &Q, K, V = T_p W^Q, T_p W^K, T_p W^V \\ &T_A = \text{UnPad}(\text{Softmax}(Mask + QK^T)V), \end{aligned} \tag{5}$$

where $\text{Pad}(\cdot)$ indicates the padding operation. W^Q, W^K, and W^V are linear weights to transform padded tokens into query Q, key K, and value V. The $Mask$ matches the attention map $A = QK^T$ in size and contains large negative values in the columns corresponding to the tokens used for padding, while other values are set to zero. The masking operation cuts off the influence of unselected tokens in the Softmax process. After a matrix multiplication operation between the post Softmax attention map and V, the tokens at padded locations are removed by the $\text{UnPad}(\cdot)$ operation, resulting in attention-enhanced tokens T_A.

After performing MS-WSA, the attention-enhanced tokens T_A sequentially undergo a sparse MLP layer and an optional context broadcasting (CB) operation [19]. The CB operation can be expressed as:

$$\text{CB}(\widetilde{T}(n)) = \frac{1}{2}\widetilde{T}(n) + \frac{1}{2N}\sum_{n=1}^{N}\widetilde{T}(n) \text{ for } n = 1, \ldots, N, \tag{6}$$

where $\widetilde{T}(n)$ represents the n_{th} output token of the sparse MLP layer. N is the total number of selected tokens. The CB operation broadcasts information among selected tokens, enriching their information density to ensure that they will be consistently selected in subsequent layers. It does not incur additional computation but leads to a looser sparsification preference for SAST. Finally, the processed tokens are scattered back to the original partitioned tokens T and reversed from windows to their original shape.

4. Experiments

Initially, we outline the experimental setup, detailing the datasets used, evaluation metrics, and various implementation specifics. Next, we compare SAST with other state-of-the-art works on two large-scale event-based object detection datasets. We also compare SAST with baseline variants and other sparsification methods. Finally, we offer insights through ablation studies, visualizations, and statistical results to thoroughly examine the effectiveness of our proposed sparsification approach.

4.1. Experimental Setup

Datasets. The 1Mpx dataset, commonly used for event-based object detection, consists of 14.65 hours of events, with a large resolution of 1280×720 pixels. It includes 7 labeled object classes. We follow previous works [15, 28, 43], utilizing 3 classes: car, pedestrian, and two-wheeler for performance comparison. This dataset has a labeling frequency of 60 Hz and contains over 25M bounding boxes. The Gen1 dataset comprises 39 hours of events. It has a smaller resolution of 304×240 pixels and contains 2 object classes. The labeling frequency for this dataset is 20 Hz.

Metrics. We use the COCO mean average precision (mAP) [32] as the main metric. To measure computational complexity, we compute the average FLOPs (Floating Point Operations Per Second) over the first 1000 samples in the test sets of 1Mpx and Gen1, referring to Sparse DETR [47]

		1Mpx		Gen1		
Methods	Backbone	mAP (%)	FLOPs (G)	mAP (%)	FLOPs (G)	Params (M)
SAM [31]	ResNet50 [18]	23.9	19.0	35.5	6.0	>20
YOLOv3_DVS [20]	Darknet-53 [46]	34.6	34.8	31.2	11.1	>60
RED [43]	ResNet50 [18]	43.0	19.0	40.0	6.0	24.1
ASTMNet [28]	VGG16 [50]	48.3	75.7	46.7	29.3	>100
AEC [41]	CSP-Darknet-53 [21]	48.4	58.2	47.0	20.9	46.5
AEC [41]	Deformable-DETR [68]	45.9	>50	44.5	>20	40.0
GET [42]	GET [42]	48.4	10.6 (6.3)	47.9	3.6 (2.2)	21.9
RVT [15]	Swin Transformer[36]	46.7	10.4 (6.6)	44.4	3.6 (2.3)	18.5
RVT [15]	ConvNeXt [38]	45.5	10.4 (6.6)	42.3	3.6 (2.3)	18.7
RVT [15] (baseline)	MaxViT [54]	47.4	10.3 (6.5)	47.2	3.5 (2.2)	18.5
Ours	SAST	48.3 (+0.9)	**5.6** (**1.8**, -72%)	47.9 (+0.7)	**2.1** (**0.8**, -64%)	18.9
Ours	SAST-CB	**48.7** (+1.3)	6.4 (2.6, -60%)	**48.2** (+1.0)	2.4 (1.1, -50%)	18.9

Table 1. Detection performance compared with state-of-the-art methods on 1Mpx and Gen1. The reported FLOPs belongs to the backbone. Values in brackets $(\cdot)$ indicate the A-FLOPs, excluding operations from convolutional layers, and is precisely the value we aim to reduce.

that calculates the FLOPs for the first 100 samples in the MS COCO dataset. Additionally, we report the average Attention-related FLOPs (A-FLOPs), which excludes the computations incurred by convolutional layers during calculation. The model's size is measured through its parameter count. We also compare the inference time (runtime) with baseline variants and different sparsification methods.

Implementation Details. We adopt the predefined train, validation, and test splits of 1Mpx and Gen1 datasets. The accumulation time of each sample is chosen as 50 ms. We also follow the dataset preprocessing methods of previous works [15, 28, 42, 43], such as removing misleading small bounding boxes and downsampling the events in the 1Mpx dataset into 640 $\times$ 360 for a fair comparison. We choose RVT-B (RVT) [15] as the baseline and apply different sparsification methods to its MaxViT backbone, while maintaining consistency in the design of other layers (FPN, Head, and LSTM). The partition strategy is the same as RVT, dividing tokens into windows and grids (a window type) in two successive SAST layers. The augmentation strategies include zoom-in, zoom-out, and horizontal flipping. We utilize eight NVIDIA TITAN Xp GPUs for training. For the testing phase, we only use one NVIDIA TITAN Xp GPU.

4.2. Experimental Results

Comparison with SOTAs. We evaluate SAST on the 1Mpx and Gen1 datasets. The results compared with state-of-the-art works are reported in Tab. 1. To facilitate distinction, we define the model as SAST-CB when the optional CB operation is utilized.

On the 1Mpx dataset, our methods outperform all Transformer-based networks, including AEC (Deformable-DETR as the backbone), GET [42], and RVT [15] using different backbones [36, 38, 54]. SAST achieves a 48.3% mAP with only 28% of the A-FLOPs of RVT. However, in comparison to all works involving pure convolutional networks, SAST falls slightly below AEC (CSP-Darknet-53 as the backbone), which utilizes a more complex event representation, introduces additional data augmentations, and has 9.4 times more FLOPs. SAST-CB achieves a superior result of 48.7% mAP, surpassing all other state-of-the-art networks with merely 11% of AEC's FLOPs and 40% of the A-FLOPs compared to RVT.

Methods	mAP (%)	A-FLOPs (G)	runtime* (ms)
RVT-T [15]	41.5	**1.8**	14.5
RVT-S [15]	44.1	3.8	15.3
RVT [15] (baseline)	47.4	6.5	16.0
AViT [61]	44.5	23.4	37.0
SViT [4]	45.3	3.3	14.3
SparseTT [11]	47.6	6.5	16.1
SAST (Ours)	**48.3**	**1.8**	19.7

Table 2. Detection performance on 1Mpx using RVT variants and different sparsification methods. SAST achieves optimal performance with the least computational expense. $*$: All the runtime is tested on one NVIDIA TITAN Xp GPU.

On the Gen1 dataset, both SAST and SAST-CB are superior to all other methods, achieving 47.9% and 48.2% mAP, which is 0.7% and 1.0% higher than RVT. Moreover, the FLOPs and A-FLOPs of SAST are only 60% and 36% of RVT, respectively. In comparison to the pure convolutional network AEC (CSP-Darknet-53 as the backbone), SAST and SAST-CB achieve 0.9% and 1.2% performance gain with just 10% and 11% of the FLOPs.

Comparison with RVT variants & sparse Transformers. As shown in Tab. 2, we further compare SAST with RVT variants and sparse Transformers on the 1Mpx dataset.

RVT variants reduce A-FLOPs significantly by scaling down the model size, at the cost of substantial performance decline. AViT [35] implements token-level sparsification based on ViT. However, the computational complexity of its global self-attention after sparsification remains high. Its A-FLOPs exceeds 260% of the baseline, with a per-

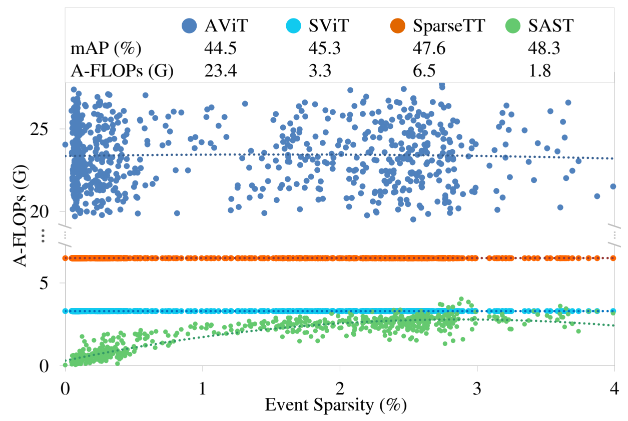

Figure 5. The computational complexity of different networks under scenes with varying event sparsity in 1Mpx. Each point represents an event sample. SAST adaptively adjusts its sparsity based on the scene complexity.

formance decrease of 2.9%. SViT [4], which builds upon window-based Swin-Transformer [36], achieves window-level sparsification at a manually set window pruning ratio (set to 50% as in the paper), lacking adaptability. Although its A-FLOPs is only 50.8% of RVT's, due to the removal of a fixed number of windows in all scenarios, its performance dropped by 2.1%. SparseTT [11] achieves a 0.2% performance gain by sparsifying the attention map. However, such mask-based sparsification does not actually reduce computational costs. Our proposed SAST, thanks to its outstanding scene-adaptive sparsification, achieves the highest mAP of 48.3%, with only 7.7% A-FLOPs of AViT and 54.5% A-FLOPs of SViT.

In Fig. 5, we illustrate the computational complexity (A-FLOPs) for the first 1000 samples in the 1Mpx dataset. It can be observed that both RVT and SViT maintain constant A-FLOPs regardless of changes in the scene. AViT shows some level of adaptability, which remains evenly distributed around its average A-FLOPs. Our proposed SAST shows remarkable scene-aware adaptability. It has very low A-FLOPs in sparse scenes while dynamically optimizing the sparsity level to maintain high performance in dense scenes.

4.3. Ablation Studies

To analyze the proposed sparse Transformer SAST, we conduct a series of ablative experiments on the 1Mpx dataset.

Scoring Method. We interchanged our scoring module with various established methods from existing sparse Transformers while maintaining consistency in other architectural designs. The scoring module of SAST, as highlighted by the superior mAP of 48.3% in Tab. 3, outperforms other scoring methods. Leveraging STP weighting across both spatial and temporal-polar domains, SAST provides a more effective and context-aware approach for evaluating token importance.

Scoring Methods	Source	mAP (%)	Params (M)
L2 Activation	SparseViT [4]	45.1	18.6
Attention Mask	SparseTT [11]	45.8	18.6
Head Scores	AS-ViT [35]	46.8	18.7
Head Importance	SPViT [24]	46.5	18.9
Scoring Module	SAST (Ours)	**48.3**	18.9

Table 3. Detection performance on 1Mpx by using different scoring methods. The scoring module of SAST surpasses other established scoring methods in sparse Transformers.

Selection Target	mAP (%)	A-FLOPs (G)	runtime (ms)
Window	46.3	3.7	20.1
Token	47.7	3.0	21.7
Window & Token	**48.3**	**1.8**	19.7

Table 4. Detection performance on 1Mpx across different selection targets. Selection on both windows and tokens forces the network to focus on the most salient and crucial features, resulting in improved performance.

Methods	Context Leakage	mAP (%)	A-FLOPs (G)
SA	Yes (batch)	40.7	16.5
S-SA	Yes (token)	44.2	20.5
MS-SA	No Leakage	46.6	21.2
WSA	N/A	N/A	N/A
S-WSA	Yes (token)	46.2	2.0
MS-WSA	No Leakage	**48.3**	**1.8**

Table 5. Detection performance on 1Mpx using different self-attention methods. MS-WSA efficiently performs WSA on selected tokens with unequal window sizes, optimizing performance by preventing context leakage between batches and tokens.

Selection Method. We compare different selection methods by training two SAST variants that only select windows and tokens, respectively. As shown in Tab. 4, applying selection to both windows and tokens achieves the highest mAP of 48.3% and the lowest A-FLOPs of $1.8G$. This strategy limits the amount of information the model can use, which forces the model to concentrate on the most salient and crucial features. Consequently, it can lead to a more efficient representation where the model learns to compress token information more densely.

Self-Attention Method. We compare different self-attention methods by applying them to selected tokens with varying window sizes. It can be seen from Tab. 5 that standard self-attention (SA) needs to process all tokens together, leading to context leakage across different batches and adversely affecting performance. Sparse Self-Attention (S-SA) employs padding to isolate tokens from different batches, but still results in context leakage between selected and padded tokens. Masked Sparse Self-Attention (MS-SA) further includes masking operation, which prevents both types of context leakage. However, as global self-attention, it still has high A-FLOPs. Window Self-Attention (WSA) can not be applied to windows of unequal size; after

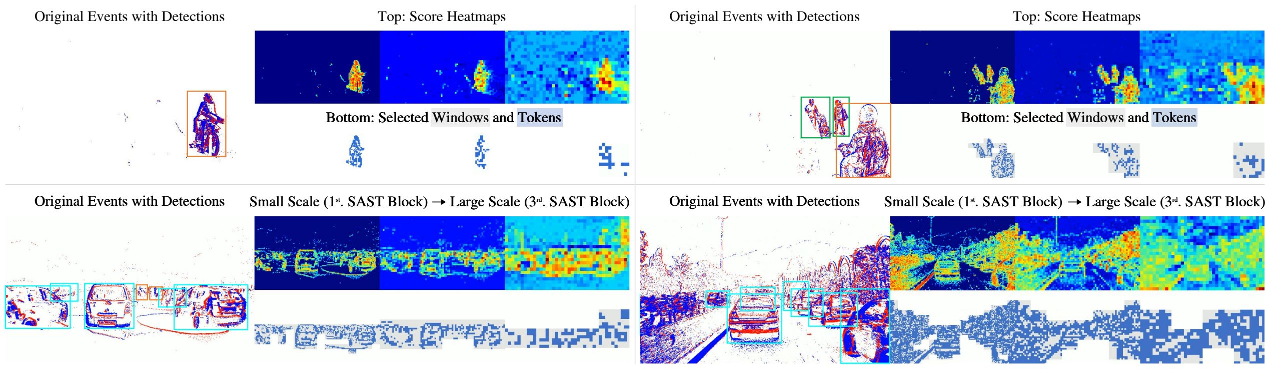
Original Events with Detections
Top: Score Heatmaps
Bottom: Selected Windows and Tokens
Original Events with Detections
Top: Score Heatmaps
Bottom: Selected Windows and Tokens
Original Events with Detections
Small Scale (1st. SAST Block) → Large Scale (3rd. SAST Block)
Original Events with Detections
Small Scale (1st. SAST Block) → Large Scale (3rd. SAST Block)

Figure 6. Visualizations of original events, score heatmaps, and selection results under four scenes in 1Mpx. As the network progresses through subsequent SAST blocks, featuring multiple downsampling stages, the scale (receptive field) of tokens expands.

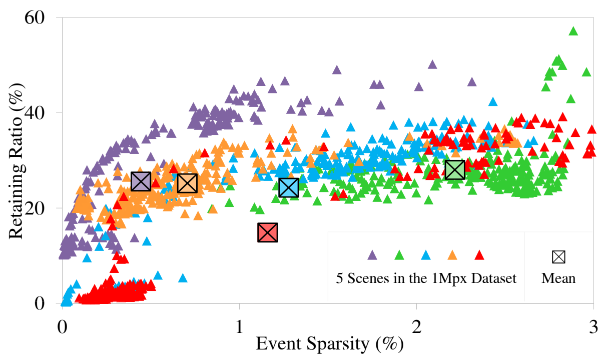

Figure 7. The averaged token retaining ratio of SAST across five scenes containing different objects in 1Mpx. Each △ corresponds to an event sample. The evident fluctuations and clusters demonstrate the model's scene-aware adaptability.

padding, Sparse Window Self-Attention (S-WSA) becomes usable but introduces context leakage between tokens, resulting in performance degradation. Only by applying MS-WSA, which isolates all context leakage and is fully parallel, is it possible to achieve optimal performance with the least computational complexity.

4.4. Adaptability Analysis

Visualizations. We train SAST on the 1Mpx dataset and infer it on four typical scenes in the test set for visualizations. The event sparsity and scene complexity increase across the four scenes progressively. In Fig. 6, we provide visualizations of the original events, score heatmaps, and the selection results of windows and tokens. For the score heatmaps and selection results, figures from left to right depict the transition from first to third SAST blocks. In a later block, the number of tokens decreases, and their scale (receptive field) enlarges. These intuitive visualizations allow us to assess the model's adaptability more comprehensively. From the visualizations of score heatmaps, we observe that the network demonstrates its scene-aware adaptability to assign important tokens higher scores. From the visualizations of window and token selection results, in complex scenes, the network proactively reduces the sparsity level to retain more important tokens.

Token Retaining Ratio. The scatter plot in Fig. 7 illustrates the relationship between the token retaining ratio and event sparsity across five scenes in the 1Mpx dataset. The general trend in the figure shows that SAST retains more tokens as event sparsity increases, demonstrating its ability to optimize the sparsity level based on the scene complexity. Furthermore, there are significant fluctuations in token retaining ratios at equivalent event sparsity. This reveals that SAST's optimization is not only based on the event sparsity but also on the distinct characteristics of each scene, such as the classes, numbers, and sizes of objects within the scene. This is also evidenced by the pronounced clustering of samples from distinct scenes, indicating that SAST can perceive the underlying patterns of different scenes, adaptively tuning its sparsity strategy accordingly. In essence, the scatter plot demonstrates that SAST can dynamically tailor its sparsity level to the specific demands of each scene, thereby exhibiting scene-aware adaptability.

5. Conclusion

In this paper, we provide a novel vision Transformer for event-based object detection, called Scene Adaptive Sparse Transformer (SAST). SAST's adaptive sparsification mechanism enables window-token co-sparsification, significantly reducing the computational overhead. Utilizing the novel scoring module, selection module, and MS-WSA, SAST showcases scene-aware adaptability, dynamically optimizing its sparsity across different scenes for high performance. Our results confirm the effectiveness of SAST, achieving state-of-the-art mAP on the 1Mpx and Gen1 datasets while maintaining remarkable computational efficiency.

6. Acknowledgement

This work was in part supported by the National Natural Science Foundation of China (NSFC) under grants 62032006 and 62021001.

References

[1] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. Asynchronous convolutional networks for object detection in neuromorphic cameras. In *CVPRW*, 2019. 2

[2] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. In *NeurIPS*, 2021. 2

[3] Xiang Chen, Hao Li, Mingqiang Li, and Jinshan Pan. Learning a sparse transformer network for effective image deraining. In *CVPR*, 2023. 2

[4] Xuanyao Chen, Zhijian Liu, Haotian Tang, Li Yi, Hang Zhao, and Song Han. Sparsevit: Revisiting activation sparsity for efficient high-resolution vision transformer. In *CVPR*, 2023. 2, 6, 7

[5] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *ICLR*, 2021. 2

[6] Loic Cordone, Benoît Miramond, and Phillipe Thierion. Object detection with spiking neural networks on automotive event data. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, 2022. 1, 2

[7] Pierre de Tournemire, Davide Nitti, Etienne Perot, Davide Migliore, and Amos. Sironi. A large scale event-based detection dataset for automotive. *arXiv preprint arXiv:2001.08499*, 2020. 2

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 2, 3

[9] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *ICCV*, 2021. 1

[10] Tobias Fischer and Michael Milford. Event-based visual place recognition with ensembles of temporal windows. *IEEE Robotics and Automation Letters*, 2020. 1

[11] Zhihong Fu, Zehua Fu, Qingjie Liu, Wenrui Cai, and Yunhong Wang. Sparsett: Visual tracking with sparse transformers. In *IJCAI*, 2022. 6, 7

[12] Guillermo Gallego, Tobi Delbrück, and Garrick Orchard, et al. Event-based vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1

[13] Shan Gao, Guangqian Guo, and C. L. Philip Chen. Event-based incremental broad learning system for object classification. In *IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019. 1

[14] Daniel Gehrig, Antonio Loquercio, Konstantinos G. Derpanis, and Davide Scaramuzza. End-to-end learning of representations for asynchronous event-based data. In *ICCV*, 2019. 1

[15] Mathias Gehrig and Davide Scaramuzza. Recurrent vision transformers for object detection with event cameras. In *CVPR*, 2023. 1, 2, 5, 6

[16] Ryuhei Hamaguchi, Yasutaka Furukawa, Masaki Onishi, and Ken Sakurada. Hierarchical neural memory network for low latency event processing. In *CVPR*, 2023. 1, 2

[17] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv: 2103.00112*, 2021. 2

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2015. 6

[19] Nam Hyeon-Woo, Kim Yu-Ji, Byeongho Heo, Doonyoon Han, Seong Joon Oh, and Tae-Hyun Oh. Scratching visual transformer's back with uniform attention. In *ICCV*, 2023. 3, 5

[20] Zhuangyi Jiang, Pengfei Xia, Kai Huang, Walter Stechele, Guang Chen, Zhenshan Bing, and Alois Knoll. Mixed frame-/event-driven fast pedestrian detection. In *ICRA*, 2019. 1, 6

[21] Glenn Jocher. ultralytics/yolov5: v6.0 - yolov5n 'nano' models, roboflow integration, tensorflow export, opencv dnn support. *Zenodo*, 2021. 6

[22] Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Joseph Hassoun, and Kurt Keutzer. Learned token pruning for transformers. *SIGKDD*, 2021. 2

[23] Taewoo Kim, Yujeong Chae, Hyun-Kurl Jang, and Kuk-Jin Yoon. Event-based video frame interpolation with cross-modal asymmetric bidirectional motion fields. In *CVPR*, 2023. 2

[24] Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Xuan Shen, Geng Yuan, Bin Ren, Hao Tang, et al. Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In *ECCV*, 2022. 2, 7

[25] Xavier Lagorce, Cédric Meyer, Sio-Hoi Ieng, David Filliat, and Ryad Benosman. Asynchronous event-based multikernel algorithm for high-speed visual features tracking. *IEEE Transactions on Neural Networks and Learning Systems*, 2015. 2

[26] Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E. Shi, and Ryad B. Benosman. Hots: A hierarchy of event-based time-surfaces for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 1, 2

[27] Junhaeng Lee, Tobi Delbrück, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 2016. 1

[28] 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*, 2022. 1, 2, 5, 6

[29] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. Localvit: Bringing locality to vision transformers. *arXiv preprint arXiv:2104.05707*, 2021. 2

[30] Yijin Li, Han Zhou, Bangbang Yang, Ye Zhang, Zhaopeng Cui, Hujun Bao, and Guofeng Zhang. Graph-based asynchronous event processing for rapid object recognition. In *ICCV*, 2021. 1

[31] Zichen Liang, Guang Chen, Zhijun Li, Peigen Liu, and Alois Knoll. Event-based object detection with lightweight spatial attention mechanism. In *The IEEE International Conference on Advanced Robotics and Mechatronics (ICARM)*, 2021. 1, 6

[32] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 5

[33] Mengyun Liu, Na Qi, Yunhui Shi, and Baocai Yin. An attention fusion network for event-based vehicle object detection. In *ICIP*, 2021. 1

[34] Qianhui Liu, Dong Xing, Huajin Tang, De Ma, and Gang Pan. Event-based action recognition using motion information and spiking neural networks. In *IJCAI*, 2021. 1

[35] Xiangcheng Liu, Tianyi Wu, and Guodong Guo. Adaptive sparse vit: Towards learnable adaptive token pruning by fully exploiting self-attention. In *CVPR*, 2022. 2, 6, 7

[36] 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. 2, 6, 7

[37] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. In *CVPR*, 2022. 2

[38] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *CVPR*, 2022. 6

[39] Diederik Paul Moeys, Federico Corradi, Emmett Kerr, Philip Vance, Gautham Das, Daniel Neil, Dermot Kerr, and Tobi Delbrück. Steering a predator robot using a mixed frame/event-driven convolutional neural network. In *The International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, 2016. 1, 2

[40] Bowen Pan, Yifan Jiang, Rameswar Panda, Zhangyang Wang, Rogério Feris, and Aude Oliva. Ia-red^2: Interpretability-aware redundancy reduction for vision transformers. In *NeurIPS*, 2021. 2

[41] Yansong Peng, Yueyi Zhang, Peilin Xiao, Xiaoyan Sun, and Feng Wu. Better and faster: Adaptive event conversion for event-based object detection. In *AAAI*, 2023. 1, 6

[42] Yansong Peng, Yueyi Zhang, Zhiwei Xiong, Xiaoyan Sun, and Feng Wu. Get: Group event transformer for event-based vision. In *ICCV*, 2023. 1, 2, 6

[43] 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, 5, 6

[44] Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. In *ICLR*, 2022. 2

[45] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *NeurIPS*, 2021. 2

[46] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. In *CVPR*, 2018. 6

[47] Byungseok Roh, JaeWoong Shin, Wuhyun Shin, and Saehoon Kim. Sparse detr: Efficient end-to-end object detection with learnable sparsity. In *ICLR*, 2022. 5

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

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

[50] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 6

[51] Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad B. Benosman. Hats: Histograms of averaged time surfaces for robust event-based object classification. *CVPR*, 2018. 1, 2

[52] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys*, 2022. 1, 2

[53] Yi Tian and J. Andrade-Cetto. Event transformer flownet for optical flow estimation. In *British Machine Vision Conference*, 2022. 2

[54] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. In *ECCV*, 2022. 2, 6

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 5

[56] Apoorv Vyas, Angelos Katharopoulos, and Franccois Fleuret. Fast transformers with clustered attention. In *NeurIPS*, 2020. 2

[57] Zuowen Wang, Yuhuang Hu, and Shih-Chii Liu. Exploiting spatial sparsity for event cameras with visual transformers. In *ICIP*, 2022. 2

[58] Wenming Weng, Yueyi Zhang, and Zhiwei Xiong. Event-based video reconstruction using transformer. *ICCV*, 2021. 2

[59] Hao Wu, Yueyi Zhang, Wenming Weng, Yongting Zhang, Zhiwei Xiong, Zhengjun Zha, Xiaoyan Sun, and Feng Wu. Training spiking neural networks with accumulated spiking flow. In *AAAI*, 2021. 1

[60] Haixu Wu, Jialong Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Flowformer: Linearizing transformers with conservation flows. In *International Conference on Machine Learning (ICML)*, 2022. 2

[61] Hongxu Yin, Arash Vahdat, Jose Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-ViT: Adaptive tokens for efficient vision transformer. In *CVPR*, 2022. 2, 6

[62] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis E.H. Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*, 2021. 2

[63] Jiqing Zhang, Xin Yang, Yingkai Fu, Xiaopeng Wei, Baocai Yin, and Bo Dong. Object tracking by jointly exploiting frame and event domain. In *ICCV*, 2021. 1

[64] Jiqing Zhang, Bo Dong, Haiwei Zhang, Jianchuan Ding, Felix Heide, Baocai Yin, and Xin Yang. Spiking transformers for event-based single object tracking. In *CVPR*, 2022. 2

[65] Zizhao Zhang, Han Zhang, Long Zhao, Ting Chen, , Sercan Ö. Arık, and Tomas Pfister. Nested hierarchical transformer: Towards accurate, data-efficient and interpretable visual understanding. In *AAAI*, 2022. 2

[66] Junwei Zhao, Shiliang Zhang, and Tiejun Huang. Transformer-based domain adaptation for event data classification. *ICASSP*, 2022. 2

[67] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised event-based learning of optical flow, depth and egomotion. In *CVPRW*, 2019. 1, 2, 3

[68] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: deformable transformers for end-to-end object detection. In *ICLR*, 2021. 6