A Event-Driven Processing

Previous studies [62–64] have demonstrated that the commercial event-driven processor (e.g., GrAI VIP [55]) can achieve latency and FPS performance comparable to embedded GPUs (e.g., Nvidia Jetson Nano [34]) on standard-trained DNNs while consuming over one order of magnitude less energy per frame [33, 53, 61] and featuring approximately 54 times smaller chip size (e.g., GrAI VIP (7.6 mm×7.6 mm) vs. Nvidia Jetson Nano (69.6 mm×45.0 mm). Moreover, the specific optimization method, event suppression, significantly improves DNN inference performance by reducing event-triggered computations and memory accesses in the network, leading to superior latency, FPS, and energy efficiency compared to embedded GPUs. In the following subsections, we expand upon Section 4 with a more detailed explanation of 1) the inherent relationship between spatial Δ-Σ suppression and its temporal counterpart (Appendix A.1), 2) how event-driven processors efficiently exploit activation sparsity (Appendix A.2), 3) the implementation of line-based suppression in event-driven convolution (Appendix A.3).

A.1 Spatial Redundancy in Images

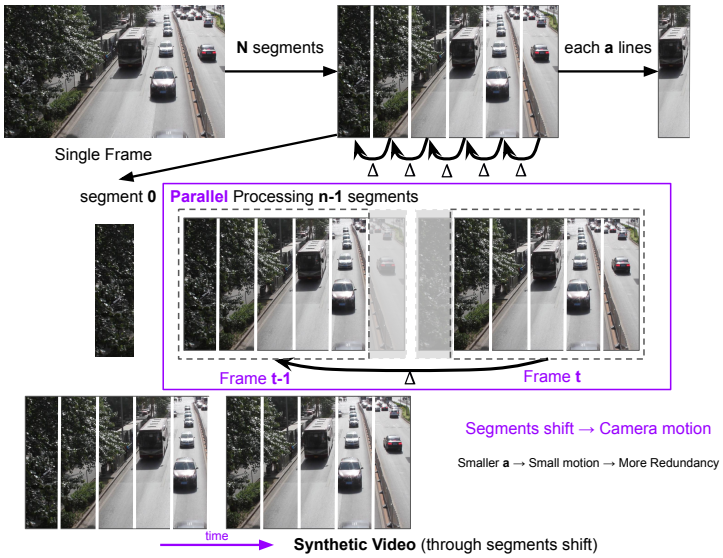


Fig. 8: Illustration of the internal relationship between ELSE and temporal suppression. Each segment represents an equally-cut tile within a given image, with no limitations in rows, columns, or blocks. The demonstrated shift is primarily horizontal but may also occur vertically.

The essence of spatial Δ-Σ suppression is to generate a 2-frame video on a given image through segment shifting, depicted in Figure 8. As shown in the purple

rectangle, instead of processing Δ-Σ modulation sequentially from the first segment to the last, we can reconstruct the $0^{th} \sim n\text{-}1^{th}$ segments as Frame_{t-1} and the $1^{th} \sim n^{th}$ segments as Frame_t, then subtracting two frames for a delta map. This delta map is equal to the delta map obtained by the sequential subtraction between the n^{th} and $n\text{-}1^{th}$ segments. We notice that the two newly generated frames exhibit a shift of one segment in their pixel positions, with each segment containing a lines. This shift can be seen as camera motion, and when it's minimal, the two frames will exhibit a strong correlation. Thus, a should be set to 1, considering that one line is the smallest component of each segment. Furthermore, as the synthetic video consists of two frames with identical values but shifted in pixel positions, the n^{th} line convolution outputs of Frame_t can be reused as the $n\text{-}1^{th}$ line outputs of Frame_{t-1}, leading to the reduction of state memory footprint. **Hence, line-based suppression can attain a sparsity level comparable to temporal suppression, while only necessitating a one-line state memory footprint for Δ-Σ modulation**. It is worth mentioning that the pixel correlation within the synthetic video not only depends on the shift a but also on the image size. For instance, a one-line shift in a 1920×2180 image results in much less motion than in a 256×512 image. More details can found in Appendix B.2.

A.2 Event-Driven Convolution

Event-Driven Architectures [12, 13, 43, 53] are a type of dataflow architectures that emulate the brain's energy and compute efficiency by executing the networks in an asynchronous, parallel and sparsity-aware event-driven manner. The event-driven convolution is, known as input-centric convolution, operated in a transposed fashion [4]. As illustrated in Fig. 9, the standard convolution reduces input values (red) via the weight kernel (yellow); the event-driven convolution broadcasts non-zero input activations (blue) via the transposed kernel (yellow). Event processing solely triggers a convolution when there's an arrival event in the input activation maps, meaning that **the entire accompanying computations and memory accesses can be skipped in the processing if the pixel stays inactive**.

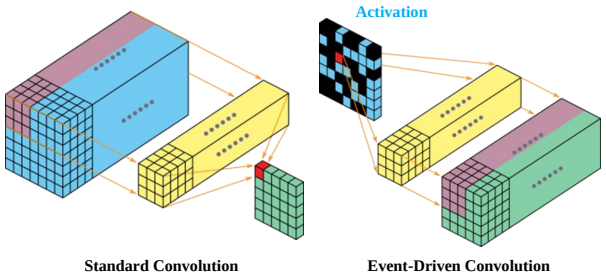


Fig. 9: Comparison between Standard Convolution (Output-Centric) and Event-Driven Convolution (Input-Centric).

Moreover, given that the event-driven convolution processes inputs in an element-wise manner, it allows convolutions to be triggered line-by-line sequentially within the input activation maps. This facilitates DNNs to execute in a deep layer fusion mode, dubbed depth-first execution [5, 23, 38, 40]. The depth-first mode promptly deallocates the utilized state memory (e.g., k_h lines) once related computations conclude, thereby only a portion of accumulation maps is stored in on-chip SRAM, as depicted in Activation Suppression of Figure 10.

A.3 Line-based Event-Driven Convolution

In DNNs, the input activation map usually represents the output of an intermediate activation layer [17, 22, 39, 46] and serves as the input to the succeeding convolution layer. Our line-based event-driven convolution initially computes element-wise differences (Δ) between adjacent activation lines and processes the convolution only on non-zero line changes. The convolution output of the Δ input line is then summed up with the one of the previous line, a process named Σ. More precisely, as shown in Figure 10, the first line of input activation map is processed directly by the convolution (green) and accumulate the outputs (Σ, blue) on the state map (grey), while the second line subtracts the corresponding values in the first line to generates the sparsified Δ line, the convolution output of Δ line is accumulated on the state of the first line and generates the equivalent convolution output of the second line, then this new output is shfited to its corresponding state position for state accumualtion. The process is repeated for the following lines until the whole input activation map is completed.

Note that our line-based event-driven convolution differs from temporal convolution in that it integrates Δ-Σ modulation across adjacent activation lines (spatially) rather than across consecutive frames (temporally). This approach offers a notable advantage by decreasing the memory usage in hardware deployment. Specifically, the spatial Δ-Σ method enables DNNs to operate in a depth-first mode, necessitating only the orange region as state, depicted in Line-based Suppression in Figure 10. This region is substantially smaller compared to the one required for Temporal Suppression.

B Supplementary Materials for Experiments

B.1 Experimental Setup

Datasets: We employ three widely-used video datasets, namely MPII [3], UA-DETRAC [56] and Cityscapes [11], as testbeds to evaluate our method ELSE across various applications, encompassing pose estimation, object detection and semantic segmentation. MPII and Cityscapes capture footage from moving cameras with distinct motion characteristics, while UA-DETRAC features recordings from static cameras for traffic surveillance. These video datasets enable the investigation of our approach in both spatial and temporal domains. Additionally, to examine the suppression performance of ELSE on single input images, we

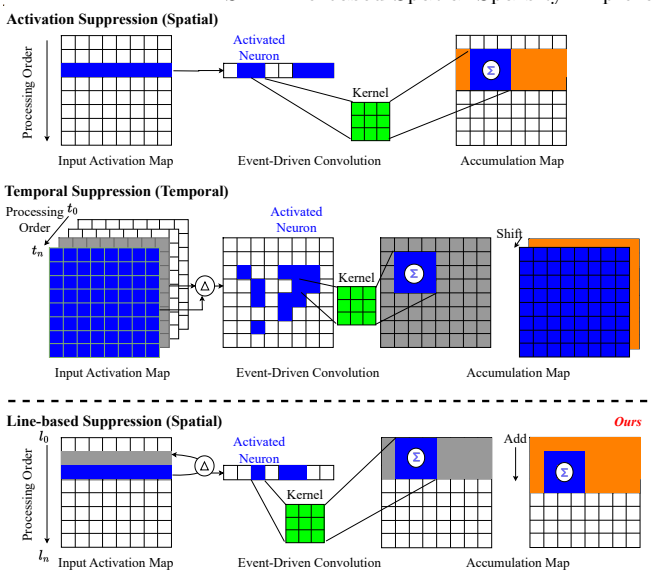


Fig. 10: Illustration of the proposed line suppression method ELSE on a convolution layer. The blue segment indicates the activated neurons at execution time t. The gray segment indicates the activated neurons at execution time $t-1$. The orange segment represents the required state, which reserves memory footprint on hardware for event-driven convolution processing.

also conduct experiments on the image datasets VOC [18], ImageNet [14], and DIV2K [1,32] for tasks such as object detection, image classification, and super resolution, respectively.

Applications: We conduct extensive event suppression experiments to showcase the effectiveness of our proposed method ELSE and its mixed-strategy variants in both high-level vision tasks, such as image classification, object detection, semantic segmentation, and human body pose estimation, as well as low-level vision tasks like super resolution.

Implementation details: The standard models (except MobileNet/ImageNet) are trained from scratch but initialized by pre-training on ImageNet [14]. All event suppression experiments have been conducted on these standard models. We utilize only half of the standard training epochs and decay the learning rate by a factor of 10 at 1/3 epoch. To ensure a fair comparison, we adhere to the same training schedules (if training is necessary). Additionally, for hardware efficiency and model deployment simplicity, we implement power-of-two values for all policy thresholds in the examined networks.

Furthermore, previous studies [21,35,61,62,64] focus solely on minimizing event numbers, neglecting variations in MACs per event, especially in modern lightweight network architectures like MobileNets and EfficientLite. To ensure fair comparison, we also weight sparsity-inducing penalties with event-triggered computations in those compared state-of-the-art (SOTA) methods, leading to superior overall MAC reduction compared to the initial implementations.

Finally, we minimize event-triggered computations (MAC) within a 0.3% relative accuracy drop. For human body pose estimation, we report the results on the validation set of MPII with input size 256×256 and use the PCK metric with a detection threshold of 0.5 [58]. For object detection, we report the results on the validation set of UA-DETRAC with input size 300×300 and the validation set of VOC(2007, 2012) with input size 640×640, using the mean Average Precision (mAP) with an IoU threshold of 0.5 in VOC format [18]. For semantic segmentation, we report the results on the validation set of Cityscapes with input size 256×512 and use the pixel intersection-over-union averaged across the 19 classes (mIoU) [10]. For super resolution, we report the results on the validation set of DIV2K [1] with 4× downscaled input images and use the peak signal-to-noise ratio(PSNR) [36].

B.2 Effect of image size on ELSE

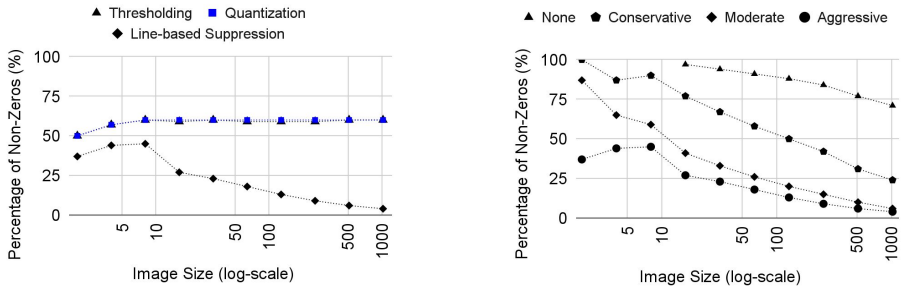
We set the same threshold value of 2^{-2} for three experiments with different event suppression methods on resized images from Cityscapes dataset [11]. Note that thresholds for quantization and line-based suppression indicate quantization scales. Figure 11a illustrates the effect of image size on the pixel suppression, where a higher percentage of non-zeros indicates fewer suppressed pixels. We observe that thresholding and quantization exhibit similar suppression performance, which even slightly decreases as the image size increases. However, our line-based suppression demonstrates additional suppression gain over the other two approaches consistently across various image scales. This gain is significantly enhanced as the image size increases. Thus, larger image sizes yield more benefits in line-based suppression, potentially resulting in comparable or even better computation savings than temporal suppression, as depicted in Figure 12.

B.3 Effect of quantization on ELSE

We experiment with different threshold values (i.e., quantization scale) for line-based suppression on resized images from Cityscapes dataset [11]. Figure 11b illustrates the effect of quantization in line-based suppression, where a higher percentage of non-zeros indicates fewer suppressed events. We observe that increasing the threshold value in quantization consistently reduces the non-zero values across images of different sizes. Additionally, the combination of a high threshold and large image size can effectively suppress events close to zero.

B.4 Layerwise comparison between ELSE and Temporal Method

The layerwise results of state memory cost and computation saving ratio of ELSE and CATS [62] are presented in Figure 12. As shown in the green boxes, we observe that ELSE achieves a comparable suppression effect to temporal suppression (Temporal/ELSE (MAC) is close to 1), while CATS requires extensive additional state memory footprint in those layers. These convolution layers



(a) Image Size vs. Percentage of Non-Zero Pixels (Fixed Threshold: 2^{-2}),

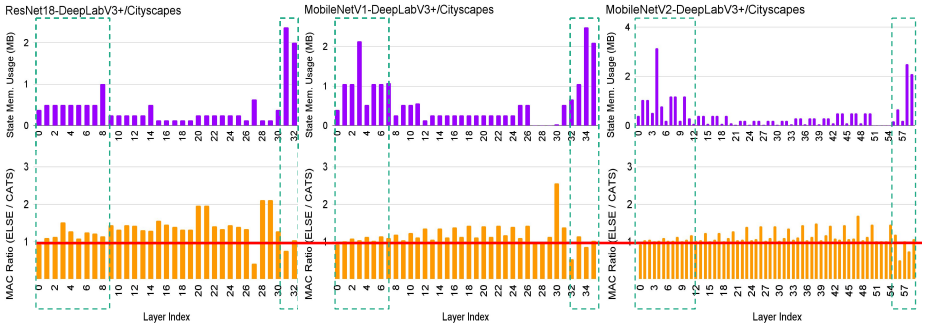
(b) Threshold vs. Percentage of Non-Zero Pixels.

Fig. 11: Study of the suppression effect of image size (a) and threshold value (b) with resized images from Cityscapes [11]. Specifically, we consider None: 2^{-5}, Conservative: 2^{-5}, Moderate: 2^{-3}, and Aggressive: 2^{-2} in (b).

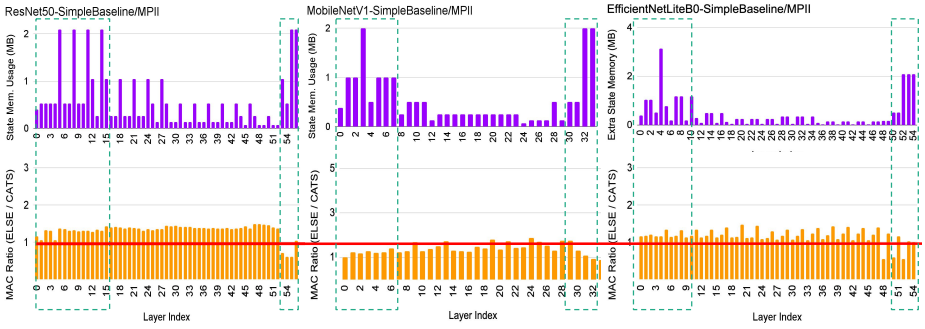
share the commonality of having large-size input activation maps, consuming serious memory footprint for temporal execution. The observation suggests that replacing temporal suppression of large-size input layers with ELSE could lead to memory savings and not compromise the achieved computation reduction. Additionally, it is worth mentioning that ELSE shares the same Δ-Σ modulation and event transmission policy (i.e., quantization) as temporal approaches, such as CATS [62], DAL [61] and SpArNet [33]. We can efficiently allocate the replaceable layers following Equation (9) and replace them without incurring any retraining costs. However, other temporal approaches [16, 45] employ different event transmission policies than ELSE. To achieve significant memory savings with these approaches, retraining is required.

B.5 Results on Non-ReLU Activation Networks

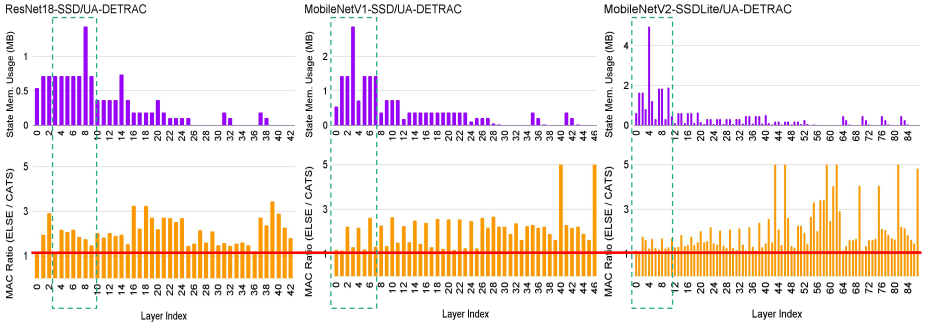
Modern deep neural networks [20,24,26,36,52] often incorporate advanced activation functions (e.g., Swish [46] and SiLU [17]), or even remove ReLU activation functions (e.g., linear activation [26, 36, 52]) to seek exceptional performance. However, these Non-ReLU Activation Networks lack natural activation sparsity, making it challenging for sparsity-inducing penalties [21, 35] to directly induce sparsity. One promising engineering approach is to simply replace those advanced activation functions with ReLU, which may result in approximately 1% accuracy drop [30, 64]. However, our method ELSE can efficiently induce sparsity in those networks without compromising accuracy. As illustrated in Figure 13a, the l_1 regularization method [21] barely reduces computations compared to the dense model, even with a sacrifice in accuracy. In contrast, ELSE can reduce computations by a factor of 1.95 and it surpasses the state-of-the-art (SOTA) thresholding method STAR [64] in cases of low accuracy drop. Furthermore, our mixed spatial approach, Mix(ELSE, STAR), integrates the layerwise strengths of ELSE and STAR, resulting in a significant computation reduction by a factor of 1.62 over ELSE.



(a) Semantic Segmentation



(b) Human Body Pose Estimation

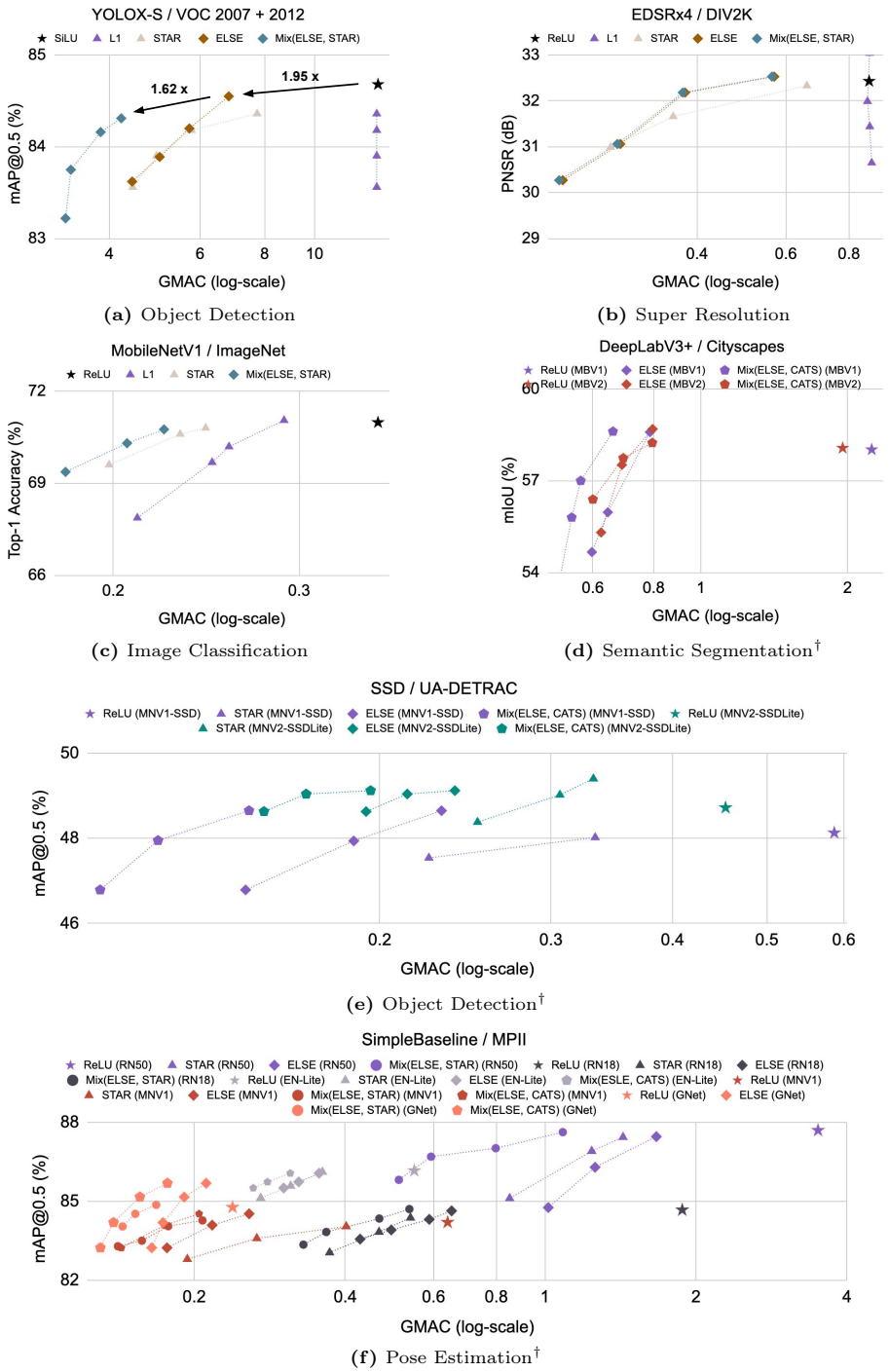


(c) Object Detection

Fig. 12: An extension to Figure 5. Layerwise state memory cost (top) and MAC ratio (bottom) between our method ELSE and a temporal suppression method CATS [62] for AI applications: semantic segmentation (a), pose estimation (b), and object detection (c). Highlighted in green boxes are layers where temporal suppression yields minimal compute reduction gains compared to ELSE, while incurring heavy memory cost.

B.6 Results on Various AI Applications

Figure 13 demonstrates accuracy/computation Pareto curves for various high-level and low-level vision applications. Adjusting the coefficients of sparsity-inducing penalties for ELSE and its mixed variants, such as Mix(ELSE, STAR) and Mix(ELSE, CATS), can reduce computational costs but may lead to increased error rates. As illustrated in Figure 13a and Figure 13c, our mixed spatial approach, Mix(ELSE, STAR), consistently outperforms the SOTA spatial approach STAR [64] on the Pareto frontier, achieving significantly lower computational resources with comparable accuracy. Furthermore, previous works [16, 45, 61, 62] demonstrate that the temporal suppression approach achieves state-of-the-art event suppression performance when a video dataset is available. To reduce the state memory footprint of CATS while maintaining its SOTA suppression performance, we replace the memory-overhead layers in CATS by our method ELSE. As a result, our spatio-temporal mixed approach, Mix(ELSE, CATS), yields additional computation savings over ELSE, achieving new Pareto frontiers across conducted experiments, as illustrated in Figure 13d, Figure 13e, and Figure 13f.



(a) Object Detection **(b)** Super Resolution

(c) Image Classification **(d)** Semantic Segmentation†

(e) Object Detection†

(f) Pose Estimation†

Fig. 13: Accuracy/computation(GMAC) trade-offs across diverse event suppression methods on different applications. GMAC denotes billion event-triggered multiply-accumulates. † indicates datasets include videos for both training and evaluation. Our proposed mixed approach consistently represent the Pareto frontier in all experiments.