A. Efficient Proxy Metric for Performance

A.1. Event-Driven Convolution

Event-driven architectures (e.g. NeuronFlow[28, 29]) are a type of dataflow architectures that emulate the brain's energy and compute efficiency by executing the networks in an asynchronous and parallel event-driven manner. As illustrated in Fig. 10, a convolution is only performed when there's an arrival event in the input activation maps (i.e., sigma maps in full-frame inference / delta maps in delta-frame inference), meaning the entire accompanying computations and memory accesses can be skipped in the processing if the neuron stays inactive.

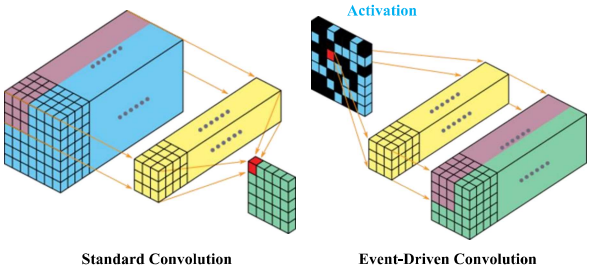

Figure 10: Performing convolution on conventional hardware versus an event-driven processor.

A.1.1 Multiply-Accumulates (MACs)

For a convolution layer, the number of filters is defined by C_{out} and their size are noted $C_{in} \times H_k \times W_k$, where C, H and W stands for channel, height, and width. The input and output of the layer are composed of a set of feature maps, with shapes $(C_{in} \times H_{in} \times W_{in})$ and $(C_{out} \times H_{out} \times W_{out})$ respectively. In the following, we consider the padding mode "same" and a stride S. The quantity of total input events of a convolution are indicated by N_{evt}, while the proportion of non-zero input events, representing density, is denoted as D_{evt}. Consider a deep neural network as a stack of L convolution blocks, each including one convolution layer (e.g., Conv2D, DepthwiseConv2D, TransposeConv2D) followed by one activation layer (e.g., ReLU [15]). The amount of multiply-accumulate (MAC) operations in the i^{th} convolution block can be described as

$$\begin{aligned} MAC_i^d &= C_{out} \times H_{out} \times W_{out} \times C_{in} \times H_k \times W_k \\ &= N_{act}/S^2 \times C_{out} \times H_k \times W_k \\ MAC_i^s &= D_{act} \times N_{act}/S^2 \times C_{out} \times H_k \times W_k, \end{aligned} \tag{16}$$

where MAC_i^s represents the MAC operations accounting for input event sparsity, while MAC_i^d does not.

A.1.2 Latency vs. Event Density

While running a deep neural network on an event-driven processor, convolution layers possess a significant portion of computes in network inference. Additionally, most event-driven architectures are non-von Neumann architectures, adopting the NMC (Near-Memory Computing[1]) technique to restore data for efficient reading and writing. This makes the total inference time T_{net} approximate the sum of the processing time T_i of the i^{th} convolution layers. Therefore, under a naive mapping strategy, the network inference time T_{net} is roughly proportional to the input event density D_{evt} of network, as shown in Eq. (17) and Eq. (18).

$$\begin{aligned} MAC_{total}^s &= \sum_i^L MAC_i^s \\ MAC_{total}^s &= \sum_i^L D_{evt_i} \times N_{evt_i}/S_i^2 \times C_{out_i} \times H_{k_i} \times W_{k_i} \end{aligned} \tag{17}$$

$$T_{net} \simeq \sum_i^L T_i \simeq MAC_{total}^s \propto \sum_i^L D_{evt_i} \Rightarrow \boldsymbol{T_{net} \propto D_{evt}} \,. \tag{18}$$

To confirm the linear correlation between latency and event density on hardware, we generate several DNN architectures from the NAS-Bench-201 search space. Our experiments cover a range of event densities, varying from 5% to 100% in 5% increments. The plots in Figure 11 demonstrate that the relationship between event density and latency is approximately linear for identical DNN architectures. However, the slopes of distinct DNN architectures differ considerably, primarily due to the varying average computations triggered by a single event across these networks.

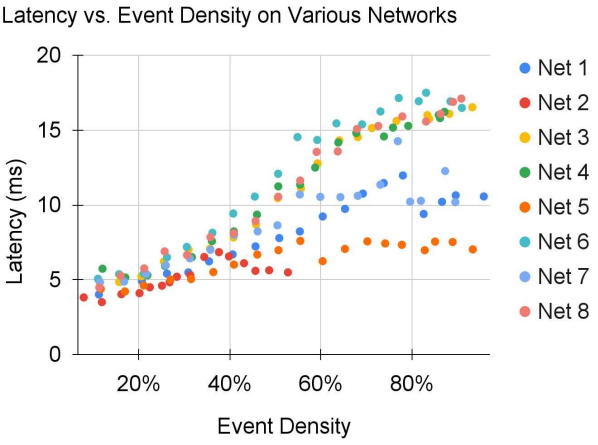

Figure 11: The relation between GrAI-VIP on-chip latency and event density across various network architectures.

A.1.3 Energy vs. Event Density

During network inference, energy consumption comprises two main parts: memory accesses and logic computation. Memory accesses can be described as data flowing from and to the memory, which consumes a significant sink of energy. For an energy estimation, we simply categorize the memory accesses that happen in each event-driven convolution block (Conv-ReLU) into three components: read operations to activation out (ReadO), read operations to parameters (ReadW), read and write operations (ReadA, WriteA) to accumulate states (Acc States), as depicted in Fig. 12.

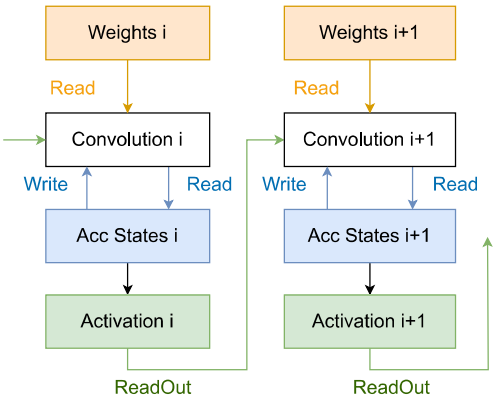

Figure 12: Memory accesses in event-driven convolution.

a. ***Read operations to event out (Green):*** ReadOut occurs before each convolution block, it reads N_{ro} times depending on the inference execution mode, i.e. $N_{ro_i} = 3$ if block i runs for delta-frame inference (temporal) and $N_{ro_i} = 1$ for full-frame inference (spatial).

$$ReadO_{i+1} = N_{ro_i} \times H_{in_{i+1}} \times W_{in_{i+1}} \times C_{in_{i+1}}. \quad (19)$$

b. ***Read operations to parameters (Orange):*** In an event-driven convolution layer $i+1$, solely non-zero out events from the former convolution layer i will trigger a read for the associated weight parameters W_{i+1}.

$$ReadW_{i+1} = C_{out_{i+1}} \times H_{k_{i+1}} \times W_{k_{i+1}} \times N_{evt_i} \times D_{evt_i}. \quad (20)$$

c. ***Read and Write operations to states (Blue):*** The output feature maps (states) from each convolution are retained for the following layer processing, or for the subsequent frame computation in the case of delta-frame inference. Therefore, the number of reading and writing states can be estimated as:

$$\begin{aligned} &ReadA_{i+1} + WriteA_{i+1} \\ &= C_{out_{i+1}} \times H_{k_{i+1}} \times W_{k_{i+1}} \times N_{evt_i} \times D_{evt_i} \times 2. \end{aligned} \quad (21)$$

Accordingly, the complete energy consumption utilized for memory accesses under fp16 execution can be formulated as:

$$E_{mem} = \sum_i^L E_{mem}^{fp16} \times (ReadO_i + ReadW_i + ReadA_i + WriteA_i), \quad (22)$$

where E_{mem}^{fp16} is the energy cost of a single read/write operation in SRAM at half-precision floating points (fp16).

Apart from the energy consumption associated with memory accesses, there is an additional energy expenditure related to multiply-accumulate (MAC) operations. We calculate the energy cost of each MAC operation by multiplying it with its respective frequency of occurrence, as per the computation outlined in Eq. (17).

$$E_{logits} = \sum_i^L (E_{add}^{fp16} + E_{mul}^{fp16}) \times MAC_i^s, \quad (23)$$

where E_{add}^{fp16} and E_{mul}^{fp16} represent the energy cost of individual addition and multiplication operations at half precision. Consequently, the energy estimation model tailored for event-driven processor can be succinctly expressed as

$$\begin{aligned} E_{net} &= E_{mem} + E_{logits} \\ &= \sum_i^L E_{mem}^{fp16} \times (ReadO_{i+1} + ReadW_i + ReadA_i + WriteA_i) \\ &+ \sum_i^L (E_{add}^{fp16} + E_{mul}^{fp16}) \times MAC_i^s \\ &\simeq C_{mem} \times \sum_i^L D_{act_i} \times N_{act_i} + C_{logits} \times \sum_i^L D_{act_i} \times N_{act_i} \\ &\Rightarrow E_{net} \propto \sum_i^L D_{act_i} \times N_{act_i} \Rightarrow \boldsymbol{E_{net} \propto D_{act}}, \end{aligned} \quad (24)$$

where C_{mem} and C_{logits} indicate the constants.Drawing from our modeling approach, we can infer that the energy consumption of network is approximately in proportion to its event density during inference.

B. Event-Driven DNN Processor GrAI-VIP

Table 2: Resources and Features of GrAI-VIP

RESOURCES/FEATURES	GRAI-VIP
PROCESS	TSMC 12FFC
SILICON AREA	$7.6 \times 7.6\, mm^2$
TRANSISTORS	$4.5\, G$
MAX # NEURON CORES	144
MAX # NEURON	18 MILLIONS
MAX # SYNAPSES	48 MILLIONS
ON-CHIP MEMORY	36 MB
INFORMATION CODING	GRADED SPIKE EVENTS (UP TO 16-BIT PAYLOAD)
PROCESSING TYPE	16-BITS FLOATING POINTS
SYNAPSES TYPE	2/4/8/16-BITS FLOATING POINTS
FREQUENCY	$650\, MHz$

In this section, we present a concise overview of the event-driven processor GrAI-VIP, which serves as a crucial component in our experiments to evaluate the hardware performance of event suppressed models. GrAI-VIP stands as a

commercially-available event-driven neural-network accelerator, building upon the successor of NeuronFlow [28, 5], and is developed by GrAI Matter Labs. As illustrated in Fig. 13, GrAI-VIP is a 12-nm taped-out System-on-Chip (SoC), comprising a grid arrangement of SIMD-4 event-driven cores in a 12×12 configuration. Each core is equipped with $2Mbits$ on-chip memory for the storage of both weights and neuron states in an energy-efficient and performant manner. Additionally, each event-driven core is furnished with a set of event queues and vector units, contributing to enhanced performance and energy efficiency. Furthermore, a comprehensive depiction of the distinctive attributes of GrAI-VIP is provided in Tab. 2, while its hardware development kits (HDK) are visually exemplified in Fig. 14.

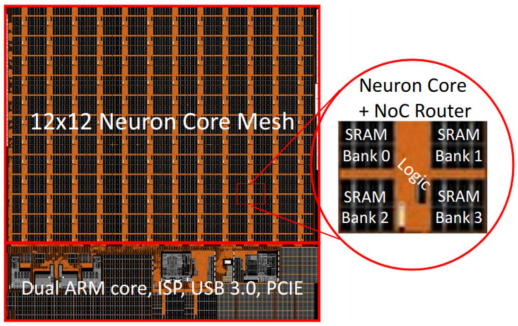

Figure 13: Block diagram of the event-driven neural-network accelerator (GrAI-VIP). The zoom-in shows the high-level structure of a neuron core.

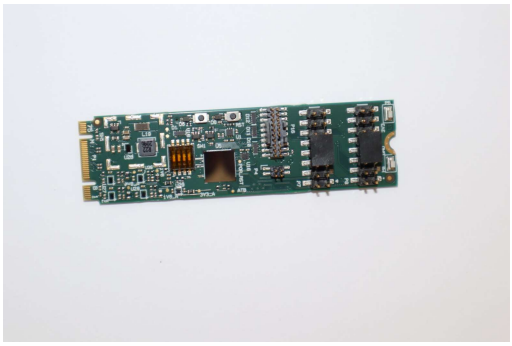

Figure 14: GrAI-VIP 80mm M.2 board.

C. Bayesian Optimization with SAT

The core of our study involves combining activation suppression and temporal suppression to achieve a cumulative effect in event suppression. Therefore, the problem can be formulated as:

$$\begin{aligned} &\min_{w,\theta} \quad L_{total}(w,\theta), \\ &\min_{w,\theta} \quad L_{task}(w,\theta) + \lambda_s L_{sigma}(w,\theta) + \lambda_d L_{delta}(w,\theta). \end{aligned} \tag{25}$$

Eq. (25) shows that the efficacy of our suppression training hinges on three components: task loss, sigma sparsity penalty, and delta sparsity penalty. However, these three loss terms compete, since excessive optimization of one may result in the suboptimal optimization of the other two. Therefore, the optimization focus is regulated by coefficient pairs (λ_s, λ_d) associated with these loss components. To streamline the training effort for various coefficient pairs, we employ Bayesian Optimization (BO) for efficient hyper-parameter search.

We provide an illustrative example of MobileNetv1-SSD utilizing the EgoHands dataset to visually elucidate the concept behind our tailor-made target function, known as the Sparsity-Accuracy Aware Target (SAT), as described in Eq. (26).

$$f(\lambda_s^i, \lambda_d^i) = S_{evt}(\lambda_s^i, \lambda_d^i) * \sigma(\beta * (C(\lambda_s^i, \lambda_d^i) - C_{lim})). \tag{26}$$

First, the top two heat maps in Fig. 15 respectively represent the accuracy and event sparsity maps of CATS-optimized models with 25 coefficient pairs (λ_s, λ_d). We observe that as λ_s and λ_d increase, the model's accuracy drops and the event sparsity increases from the left-bottom to the right-top. Our optimization objective is to maximize the event sparsity S_{evt} within the network while maintaining the model accuracy C above a quality constraint of C_{lim}. Assuming the standard-trained model has an accuracy of 95.35%, and we aim to preserve 99.5% of the model's quality after optimization. Thus, the quality constraint C_{lim} is set at 94.87%. To account for both metrics (accuracy and event sparsity) in optimization, we generate a soft mask through the sigmoid function $\sigma(\beta * (C(\lambda_s^i, \lambda_d^i) - C_{lim}))$ based on accuracy (where $\beta = 10^3$), as shown in the left-bottom heat map of Fig. 15. This mask selects a few trials that meet the quality constraint. Subsequently, we multiply this masked accuracy map with the event sparsity map to visualize our target function SAT in the right-bottom heat map of Fig. 15. As a result, the peak score (indicated in red circle) in the target function map represents the optimal CATS-optimized model under the given quality constraint. In general, it only takes 5 to 7 trials with different coefficient pairs to reach this optimum, leading to a reduction in training time by over $4\times$. This confirms the effectiveness of SAT in Bayesian Optimization.

D. Static camera vs. Moving Camera

Table 3 presents the outcomes of event suppression using the FairMOT-yolov5s model on the MOT17 dataset for object tracking with both static and moving cameras. One noteworthy observation is that temporal suppression (temporal) from static cameras achieves a $1.30\times$ lower event density and a $1.77\times$ lower standard deviation in the temporal domain compared to that from moving cameras. This finding reveals the higher stability of network computations

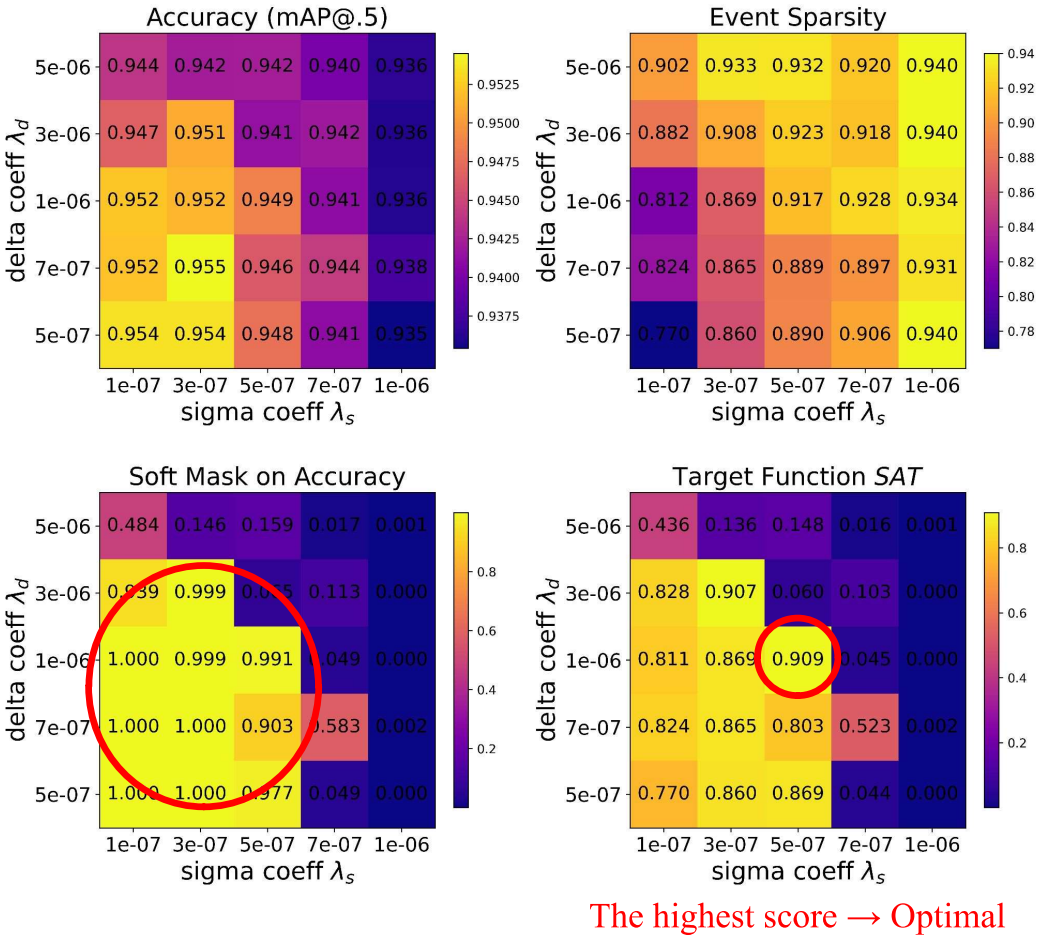

Figure 15: The visualization of our target function SAT.

Table 3: Event suppression results of FairMOT on MOT17 (3 videos recorded from static cameras: "MOT17-02-SDP", "MOT17-05-SDP", "MOT17-09-SDP", 4 videos recorded from moving cameras: "MOT17-05-SDP", "MOT17-09-SDP", "MOT17-11-SDP", "MOT17-13-SDP"). $D^s_{ev}*$ represents the event density in the spatial domain during full-frame inference, while $D^d_{ev}**$ denotes the event density in the temporal domain during delta-frame inference.

Model	Static Camera		Moving Camera	
	D^s_{ev}*[%] mean/std	D^d_{ev}**[%] mean/std	D^s_{ev}*[%] mean/std	D^d_{ev}**[%] mean/std
Baseline	48.95 / 0.09	50.34 / 0.55	49.01 / 0.26	52.44 / 1.07
Activation	34.82 / 0.06	35.99 / 0.44	34.69 / 0.20	37.39 / 0.73
Temporal	38.93 / 0.05	22.47 / 1.08	38.73 / 0.19	29.15 / 1.91
CATS	34.01 / 0.03	19.82 / 0.95	33.76 / 0.16	25.81 / 1.69

in videos with less motion.

The second noticeable trend is that activation suppression (activation) results in nearly identical event density across random frames. Remarkably, it can reduce the standard deviation of event density even below the baseline level.

A third observation is that combining activation suppression with temporal suppression mitigates the standard deviation of event density in both moving and static camera scenarios, resulting in improved stability of event suppression on top of temporal suppression. Despite the standard deviation of CATS event densities remaining below 2% across frames, indicating high temporal stability, it's important to note that variations might arise in different dataset scenarios. Further evaluation across various scenarios is necessary for a comprehensive understanding.