ELSE: Efficient Deep Neural Network Inference through Line-based Sparsity Exploration

Zeqi Zhu[1,2], Alberto Garcia-Ortiz[2,3], Luc Waeijen[1], Egor Bondarev[2], Arash Pourtaherian[1], and Orlando Moreira[1]

[1] Snap Inc.
[2] Eindhoven University of Technology, Netherlands
[3] University of Bremen, Germany

Abstract. Brain-inspired computer architecture facilitates low-power, low-latency deep neural network inference for embedded AI applications. The hardware performance crucially hinges on the quantity of non-zero activations (i.e., events) during inference. Thus, we propose a novel event suppression method, dubbed **ELSE**, which enhances inference **E**fficiency via **L**ine-based **S**parsity **E**xploration. Specifically, it exploits spatial correlation between adjacent lines in activation maps to reduce network events. ELSE reduces event-triggered computations by 3.14∼6.49× for object detection and by 2.43∼5.75× for pose estimation across various network architectures compared to conventional processing. Additionally, we show that combining ELSE with other event suppression methods can either significantly enhance computation savings for spatial suppression or reduce state memory footprint by $> 2\times$ for temporal suppression. The latter alleviates the challenge of temporal execution exceeding the resource constraints of real-world embedded platforms. These results highlight ELSE's significant event suppression ability and its capacity to deliver complementary performance enhancements for SOTA methods.

Keywords: efficient neural networks · activation sparsity exploration

1 Introduction

Deep neural networks (DNNs) have demonstrated exceptional performance on accuracy and efficiency across a spectrum of vision AI benchmarks. Nonetheless, the substantial computational requirements for DNN inference contribute to elevated serving costs (e.g., latency, energy and area) on hardware, particularly evident in edge applications, such as smart wearable sensors and mobile devices.

Novel AI computing architectures are emerging to tackle these challenges. Inspired by human brain's efficiency, an innovative paradigm, event-driven computer architecture, has garnered significant attention from both academia [15, 19, 28, 43, 49, 53] and industry [2, 7, 12, 13, 55]. A fundamental distinction between event-driven processors and conventional hardware (e.g., GPUs, CPUs) lies in the way of zero value processing. More precisely, conventional hardware processes all activations by default. Despite software support, zeros cannot be efficiently

discarded. In contrast, event-driven processors are designed to efficiently handle dynamic sparse activation patterns, solely triggering memory accesses and multiply-accumulate (MAC) operations only for non-zero values.

Previous studies [61, 62, 64] have demonstrated that suppressing events can roughly proportionally reduce latency and conserve energy on real-world embedded platforms. Consequently, devising an efficient event-suppression technique is crucial for improving the overall efficiency of event-driven processing.

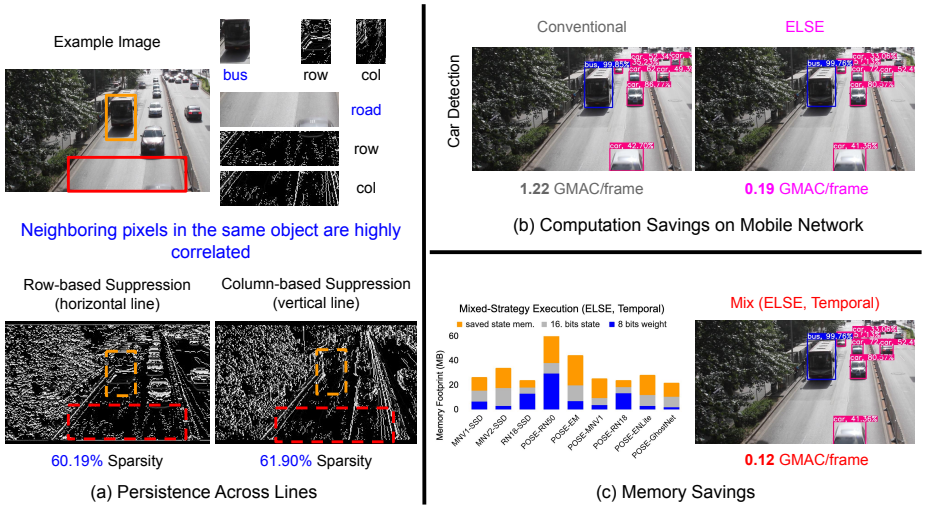


Fig. 1: Sparsity Exploration via Line-based Suppression. (a) High spatial correlation among neighboring pixels within the same object is leveraged for substantial sparsity through line-based suppression. (b) ELSE demonstrates notable computation savings even in mobile networks while maintaining high accuracy. (c) A spatio-temporal mixed approach, Mix(ELSE, Temporal [62]), yields substantial computation and memory footprint savings, enabling DNN deployment on resource-constrained embedded platforms.

In this paper, we propose a line-based event suppression method, Line-based Sparsity Exploration (ELSE), aimed at reducing computations in DNN inference. Notably, "line" functions as an abstract representation, encompassing rows or columns. Our approach is motivated by the observation of significant spatial correlation between adjacent lines within an image, where their subtraction yields a large amount of zeros. Specifically, images contain redundant spatial information, with neighboring pixels often representing the same object (e.g., bus, road) and exhibiting the same values post quantization, as shown in Figure 1.

In event-driven processing, convolution can be sequentially triggered line-by-line, enabling us to implement Δ-Σ modulation [6,44,60] on activation lines to focus solely on processing the discrete changes between them (see Appendix A). We empirically observe that 1) quantization reduces activation resolution, enhancing spatial correlation and minimizing activation changes; 2) training with sparsity-inducing penalties decreases non-zero values while maintaining accuracy [21,35]. Thus, we combine quantized Δ-Σ modulation with a training approach that

penalizes event occurrences in both activation maps and their corresponding line-based differential maps simultaneously, aiming to encourage event sparsity in subsequent convolution inputs. Overall, our method, ELSE, achieves 2.43 $\sim$ 6.49$\times$ event-triggered computation savings compared to conventional processing.

ELSE is not the sole method for exploring sparsity in the spatial domain. In recent years, various activation suppression approaches [21, 35, 63, 64] have been proposed to suppress events in activation maps, leading to significant latency and power savings for optimized models on event-driven processors. While ELSE shares the objective of exploiting spatial redundancy with these methods, it can induce additional sparsity on top of them, particularly when feature maps exhibit strong line-based correlation.

Moreover, the idea of leveraging spatial line correlation draws inspiration from temporal suppression methods [9, 16, 44, 45, 47, 61, 62]. However, to harness the computational reduction facilitated by Δ-Σ in temporal processing, the temporal approach necessitates preserving the complete activation maps (state) from time t-1 to calculate activation changes at time t, resulting in a considerable increase in memory footprint [9,16]. This storage demand is particularly burdensome for embedded devices. For example, MNV2-SSDLite [48] requires 31 MB memory for 16 bit states, exceeding the equipped on-chip SRAM of most event-driven processors [12,13,15,28,41,43] (see Figure 3). Storing the state in off-chip DRAM, while a potential solution, incurs approximately 100$\times$ more energy cost than on-chip access [29,50]. Since ELSE adopts the depth-first inference solution [5,23,40, 54, 59] to circumvent writing the complete intermediate feature maps in on-chip memory, employing a layerwise mixed approach with ELSE and state-of-the-art (SOTA) temporal methods can reduce the temporal state memory footprint by up to 2 $\sim$ 4$\times$, while maintaining computational efficiency at the temporal level.

Extensive experiments have been conducted to demonstrate the effectiveness of ELSE in reducing the event-triggered computations (MAC). Its combination with other prominent event suppression methods further addresses the challenges in enhancing computation savings for spatial methods and reducing state memory consumption for temporal methods.

The contribution of this work is three-fold:

1. We propose a novel event suppression training method ELSE by exploiting spatial correlation between adjacent lines in DNN activation maps.
2. We develop a layerwise mixed approach combining ELSE with the spatial method STAR [64] to enhance sparsity for depth-first DNN inference.
3. We introduce ELSE to complement temporal suppression methods, addressing the challenge of the overwhelming memory footprint associated with large-volume states (i.e., feature maps) for temporal DNN inference.

The remainder of this paper is structured as follow: Section 2 presents an overview of the related work. Section 3 discusses the motivations behind this paper. Section 4 provides a detailed description of ELSE and its mixed-strategy variants. Section 5 shows the experimental results. Finally, Section 6 concludes the paper.

2 Related Work

Activation Suppression aims to reduce the number of non-zero activations by exploiting spatial redundancy. Georgiadis et al. [21] introduce an L_1 sparsity-inducing penalty on activations and update weights through penalty gradients to induce activation sparsity. Kurtz et al. [35] go beyond L_1 and investigate the Square-Hoyer for sparsity exploration. In a similar vein, Zhu et al. [63] employs an adaptive training schedule to incrementally adjust the sparsity-inducing penalty during training. Recently, Zhu et al. [64] improves the accuarcy/computation tradeoffs by solely penalizing and thresholding low-magnitude activations. Previous research has encountered limitations in sparsity enhancement, stating that more aggressive suppression may pose a risk of irrecoverable drops in accuracy.

Temporal suppression decreases the event firing of repetitive locality contents in the adjacent video frames. Habibian et al. (Skip-Conv) [25] truncate small values in the activation differences between frames to enhance temporal sparsity but require frequent re-initialization to sustain model accuracy. Parger et al. (DeltaCNN) [45] and Dutson et al. (EvNets) [16] both address this issue by incorporating long-term changes through Δ-Σ modulation. However, this mechanism requires additional memory to store the long-term changes and the complete activation maps, posing high memory requirements on the resource-constrained embedded devices. Additionally, O'Connor et al. (QSD) [44] propose to quantize neuron activation changes for temporal redundancy. Yousefzadeh et al. (DAL) [61] further employ a sparsity penalty on the adjacent activation differences for temporal sparsity enhancement. Zhu et al. (CATS) [62] suggest to simultaneously enhance the activation and temporal sparsity for a synergistic suppression outcome. These three methods do not necessitate the storage of long-term changes, yet they still require retaining the complete activation maps from the previous timestamp at runtime. On the contrary, our method, ELSE, leverages Δ-Σ modulation to achieve temporal sparsity in the spatial domain, reducing the memory footprint from a complete activation map to a few activation lines using depth-first convolution execution [5]. This approach largely decreases memory usage, facilitating deployment on event-driven embedded platforms.

Mix-strategy Suppression. Activation suppression constrains the hardware performance by inadequately inducing sparsity, while temporal suppression costs additional state memory, also limiting the performance. Yousefzadeh et al. [61] have recognized the substantial memory footprint required for temporal state storage and have proposed the selective use of temporal Δ-Σ modulation for specific layers within the network, which leads to significant sparsity degradation. However, our empirical findings reveal that ELSE complements temporal suppression by substituting high-memory cost temporal layers, achieving comparable sparsity with a significantly reduced memory footprint.

3 Motivation

3.1 Proportionally Enhanced Performance via Event Reduction

Event-driven architectures mimic the brain biological principles by featuring memory-processor co-localization, sparsity exploitation, and dataflow processing [2, 12, 13, 15, 19, 28, 41–43, 49, 53]. In event-driven processing, events, known as non-zero activations or activation changes, access weight memory and trigger heavy computations. Suppressing an event results in the removal of its associated memory accesses and computations on hardware. The approximated linear relationship between event density and on-chip latency in real hardware is evident in Figure 2. Hence, our first motivation is to devise an optimization technique that intentionally suppresses events within the inputs of computationally-intensive layers (e.g., Conv2D), reducing the overall network MAC computations.

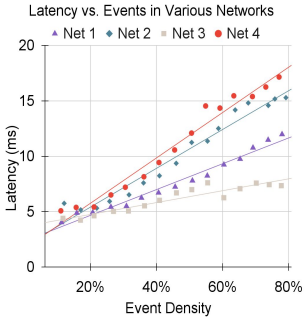


Fig. 2: Relation between event density and run-time latency across various DNNs on an event-driven platform GrAI-VIP [43].

3.2 On-chip Memory Constraints for Temporal Event Suppression

To harness the redundancy in temporally correlated data, retaining full-size feature maps from the previous timestamp during inference is crucial. However, as shown in Figure 3 (right), the state memory dominates the memory consumption, which often exceeds the on-chip limits (3∼26 MB) of the available event-driven processors [7, 12, 13, 15, 28, 42, 43], thereby hindering hardware deployment and high-performance dataflow mapping. Thus, our second motivation is to develop a suppression method that preserves the attained sparsity and accuracy of temporal methods, while concurrently diminishing the state memory footprint during temporal execution.

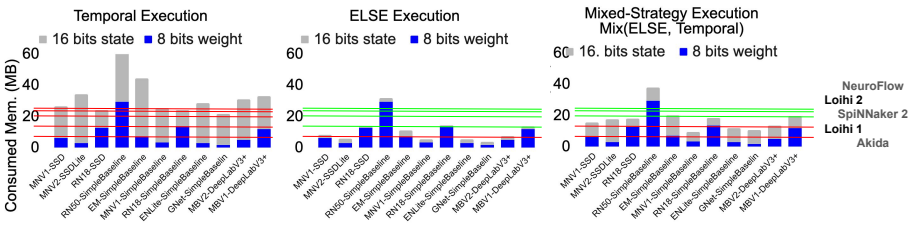


Fig. 3: Network memory footprint estimation vs. on-chip memory constraint of event-driven processors. The green line represents that over half of the examined networks meet the target hardware's memory constraints. ELSE execution (middle) and its mixed-use with temporal execution (right) reduce network state memory compared to temporal execution alone (left), enabling mapping on most event-driven processors.

4 Proposed Method

In this section, we elaborate on our proposed method ELSE, addressing three key questions: 1) how to implement Δ-Σ modulation spatially for event suppres-

sion; 2) how to enhance line sparsity exploration through training; and 3) how to integrate ELSE with other suppression methods for improved sparsity and memory efficiency.

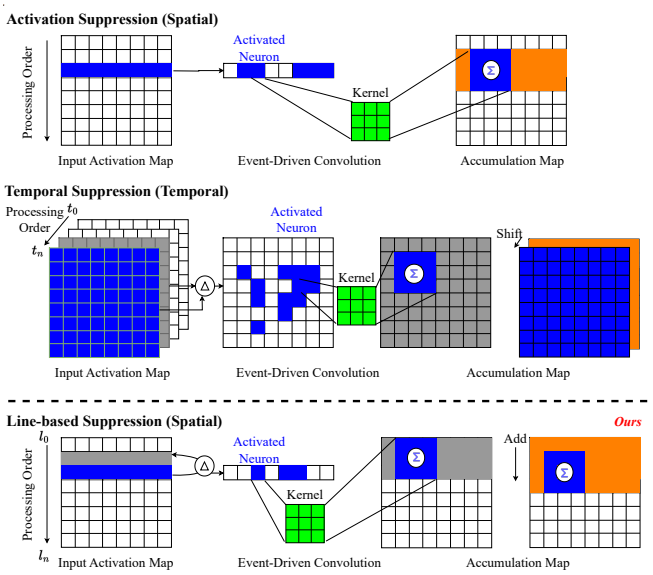


Fig. 4: Illustration of the proposed line suppression method ELSE on a convolution layer. The blue segment indicates the activated neurons at execution time t. The gray segment indicates the activated neurons at execution time t-1. The orange segment represents the required state, which reserves memory footprint on hardware for event-driven convolution processing.

4.1 Δ-Σ Line Suppression

Figure 4 illustrates the abstract execution flow of three suppression methods on the event-driven processor. In DNNs, the input activation map represents the output of an intermediate activation layer [17, 22, 39, 46] and serves as the input to the succeeding convolution layer. In contrast to activation suppression [21, 35, 63], which directly initiate computations based on non-zero activations (blue), line-based suppression initially computes element-wise differences (Δ) between adjacent activation lines, which induces more zeros due to the strong spatial correlation. On the other hand, our line-based suppression differs from temporal suppression by integrating Δ-Σ modulation across adjacent activation lines rather than consecutive frames. As depth-first DNN execution promptly deallocates the utilized state memory (orange) once the related computations conclude, only a portion of accumulation map (e.g., $mem_l^s = k_h \times l_w \times ch$, where k_h, l_w, ch represent the kernel height, line width, and feature channels, respectively.) is stored on chip for one line computation. This storage incurs negligible memory consumption compared to temporal suppression, as evidenced by $mem_l^s/mem_f^s = (k_h \times l_w \times ch)/(f_h \times f_w \times ch) = f_h/k_h \sim 0$, if $f_h >> k_h$, where f is feature map. Hence, our line-based method leverages both sparsity exploration

and memory consumption in the design space. To our knowledge, we are the first to investigate the Δ-Σ suppression in the spatial domain.

4.2 Quantized Δ-Σ Network

Consider a deep neural network as a stack of N convolution blocks, each including 1 convolution layer and 1 activation layer. Given a linear function g (e.g., a convolution) with a kernel $w \in \mathbb{R}^{c_o \times c_i \times k_h \times k_w}$ and an input activation map $x \in \mathbb{R}^{c_i \times f_h \times f_w}$, the output state map after convolution $z \in \mathbb{R}^{c_o \times f_h \times f_w}$ is computed for each frame as $z = g(x) = w * x$, where $*$ represents the convolution operation. Given the input activation map x, such that $x(i,j)$ represents the value of the activation with column coordinate i and row coordinate j in the input activation map, the Δ operation yields element-wise differences between respective pairs of lines, highlighting the changes between them. The differential feature map $d \in \mathbb{R}^{c_i \times f_h \times f_w}$ through line suppression can be described as:

$$d(i,j) = \begin{cases} x(i,0), & j = 0 \\ x(i,j) - x(i,j-1), & j > 0 \end{cases} \tag{1}$$

To obtain the values of the transformed differential feature map d, the convolution $z^d = g(d) = w * d$ is performed. We use $z^d[j]$ to represent the convolution outputs of the j^{th} line input $d[j]$, then the accumulation operation involves adding values from the the transformed differential line $z^d[j]$ to the corresponding values of previously processed lines $z[j-1]$. This Σ process effectively integrates the changes to "reconstruct" the equivalent outputs as the standard convolution.

$$z[j] = \begin{cases} z[0], & j = 0 \\ z^d[j] + z[j-1], & j > 0 \end{cases} \tag{2}$$

We use the distributive property of the linear transformation, e.g. the convolution g, the convolution output $z[j]$ of the j^{th} line input $x[j]$ can be obtained:

$$z[j] = g(x[j]) = g(x[j] - x[j-1] + x[j-1]) = g(d[j]) + g(x[j-1]) = z^d[j] + z[j-1] \tag{3}$$

Substituting Equation (1) into Equation (3) yields Equation (2), meaning the outcomes of standard inference and Δ-Σ line-based suppressed inference remain identical, effectively preventing the accumulation of errors across layers. This capability streamlines the hardware deployment process.

Due to the robust spatial correlation between adjacent lines within the same feature map, $d[j]$ tends to display sparsity. Prior research [44, 61, 62] illustrates the efficacy of quantized activation maps in fostering sparsity for temporal suppression, a phenomenon also observed in enhancing spatial correlation between neighboring lines. To quantize the activation map with a threshold $t \in \mathbb{R}^{c_i \times f_h \times f_w}$, the following applies to Equation (1) to generate the differential map $d^q \in \mathbb{R}^{c_i \times f_h \times f_w}$ for the convolution:

$$d^q(i,j) = \begin{cases} \lfloor x(i,0)/t \rceil * t, & j = 0 \\ \lfloor x(i,j)/t \rceil * t - \lfloor x(i,j-1)/t \rceil * t, & j > 0 \end{cases} \tag{4}$$

notably, replacing d with d^q in Equation (3), the equation still holds.

4.3 Event Suppression Training

Training plays a pivotal role in enhancing the suppression performance of our method ELSE and enables exploration of superior accuracy/computation trade-offs. Therefore, we formulate the event suppression problem as:

$$\begin{aligned} &\min_{w,t} \quad \mathcal{L}_{total}(w,t) \\ &\min_{w,t} \quad \mathcal{L}_{task}(w,t) + \beta_a \mathcal{L}_{act}(w,t) + \beta_d \mathcal{L}_{delta}(w,t) \end{aligned} \tag{5}$$

, where w embodies the network parameters and t represents the spatial thresholds, acting as the scale factor in quantization. $\mathcal{L}_{task}$ stands for the task loss, $\mathcal{L}_{act}$ and $\mathcal{L}_{delta}$ indicates the sparsity-inducing penalties applied on the quantized activation maps and the corresponding line-based differential maps, respectively. β_a and β_d are the coefficients of $\mathcal{L}_{act}$ and $\mathcal{L}_{delta}$ used to balance sparsity exploration and accuracy recovery in network learning.

The post-convolution state maps of training example n and layer $l \in \{1, ..., L\}$ are defined by $z_{l,n} \in \mathbb{R}^{c_o \times f_h \times f_w}$. For simplification, we apply l_1 [21] as the regularization term for sparsity-inducing purpose. Thus, $\mathcal{L}_{act}$ and $\mathcal{L}_{delta}$ can be described as follows:

$$\begin{aligned} \mathcal{L}_{act}(w_1, ..., w_L, t_1, ..., t_L) &= \frac{1}{N} \sum_{n=1}^{N} \sum_{l=1}^{L} m_l || \lfloor f(z_{l,n}(w_l, \phi_l))/t_l \rceil * t_l ||_1 \\ \mathcal{L}_{delta}(w_1, ..., w_L, t_1, ..., t_L) &= \frac{1}{N} \sum_{n=1}^{N} \sum_{l=1}^{L} m_l \sum_{r=2}^{R_l} || \Delta_r(\lfloor f(z_{l,n}(w_l))/t_l \rceil * t_l) ||_1 \end{aligned} \tag{6}$$

, where f represents the activation function [17,22,39,46] and N, L, R_l represents the amount of training examples, the amount of network layers, and the number of lines in the l^{th} layer activation maps. We also notice the layerwise computation and memory access are highly imbalanced, especially in lightweight networks [24, 31, 48, 51]. Thus, we weight the sparsity-inducing penalties with layerwise MAC m_l to optimize for minimal computations across the entire network.

4.4 Mixed Mapping with Temporal Suppression Mix(ELSE, $\phi^\dagger$)

Figure 5 shows that layers with large feature size account for over 60% of total state memory in temporal Δ-Σ execution. However, once checking the green regions, the computation ratio between ELSE and CATS [62] in these layers, $\text{MAC}_l^{ELSE}/\text{MAC}_l^{\phi^\dagger}$, is close to 1 (where a ratio over 1 indicates more event suppression in temporal; vice versa). Temporal methods $\phi^\dagger$ tend to over-consume state memory in layers where ELSE explores significant sparsity. Thus, applying ELSE in these layers can reduce memory usage without increasing computations.

Previous study [61] also claims that it typically requires storing $l + 1$ states for temporal execution for l consecutive layers. If l is sufficiently large, the number of states closely approximates the number of layers. Thus, we propose to select the temporal conversion in a module-wise manner rather than a layer-wise. We segment a network into C modules M_c based on layerwise feature map

size $f_h \times f_w$. All layers within each module are subjected to one method ψ. Through Equation (7) and Equation (8), we calculate the MAC computations and the approximate state memory consumption of each module M_c based on its assigned method. The amount of MACs in c^{th} module is computed as

$$m_c^{(\psi)} = \sum_{l=0}^{L_c} m_l^{(\psi)} = \sum_{l=0}^{L_c} e_l \times c_{o_l} \times f_{h_l} \times f_{w_l} \times p_{e_l}^{(\psi)} \tag{7}$$

, where c denotes the module index, L_c is the number of convolution layers in module M_c, ψ is the applied suppression method, and m_l, e_l, c_{o_l}, f_{h_l}, f_{w_l}, and $p_{e_l}^{(\psi)}$ represent the MAC operations, neuron count, layer output channel, layer output height, layer output width, and the percentage of activated neurons (events) at the l^{th} layer, respectively. If ψ is static, $p_{e_l}^{(\psi)}$ is set to 100%.
The state memory footprint in c^{th} module is computed as

$$s_c^{(\psi)} \approx \sum_{l=0}^{L_c} s_l^{(\psi)} = \sum_{l=0}^{L_c} c_{o_l} \times f_l^{(\psi)} \times f_{w_l} \tag{8}$$

, where $s_l^{(\psi)}$ represents the required state memory at the l^{th} layer for the hardware execution of method ψ. Meanwhile, $f_l^{(\psi)}$ denotes the required number of line storage for ψ. Specifically, if ψ denotes an activation suppression method, then $f_l^{\psi} = k_{h_l}$; if ψ represents ELSE, $f_l^{\psi} = k_{h_l} + 1$; and if ψ indicates a temporal suppression method, then $f_l^{\psi} = f_{h_l}$. To get the configuration of block-wise method ψ for a minimal sparse network computations under given state memory constraints τ, we define a linear programming problem as follows:

$$\begin{aligned} Objective: &\min_{\alpha} \sum_{c}^{C} (\alpha_c m_c^{ELSE} + (1-\alpha_c) m_c^{\phi^\dagger}) \\ Constraints: &\sum_{c=0}^{C} (\alpha_c s_c^{ELSE} + (1-\alpha_c) s_c^{\phi^\dagger}) \leq \tau \end{aligned} \tag{9}$$

, where $\phi^\dagger$ represents a chosen temporal suppression method to be combined with our method, ELSE. α is a binary variable, either 0 or 1, where 1 indicates the

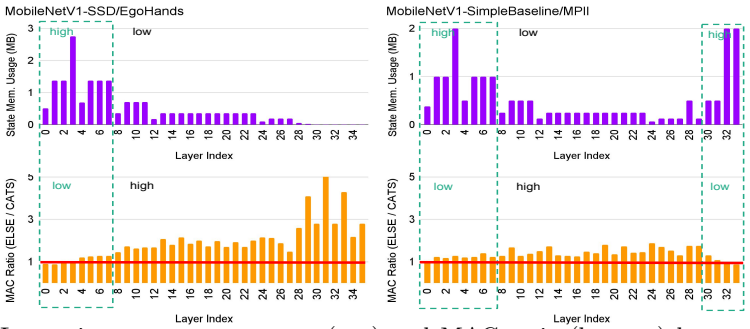


Fig. 5: Layerwise state memory cost (top) and MAC ratio (bottom) between ELSE and the temporal suppression method CATS [62] for object detection (left) and pose estimation (right). Highlighted in green boxes are the layers where temporal suppression yields minimal compute gains compared to ELSE, while incurring heavy memory cost.

application of ELSE, and 0 denotes the implementation of the other temporal method, $\phi^{\dagger}$. Minimizing the objective function involves transitioning the layers in the module towards more event-suppressed execution.

4.5 Mixed Mapping with Activation Suppression Mix(ELSE, ϕ)

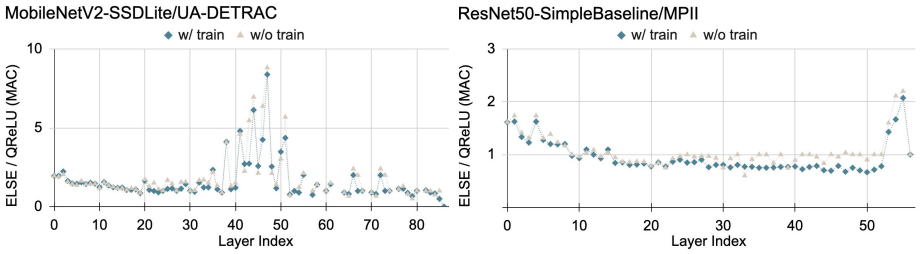


Fig. 6: Layerwise MAC reduction of ELSE versus Quantized-ReLU with and without retraining for object detection (left) and pose estimation (right).

In addition to temporal suppression, ELSE can significantly enhance the activation suppression effect by $2 \sim 8\times$ in layers with heavy line-based redundancy, as depicted in Figure 6. Furthermore, we observe that the layerwise MAC ratio of ELSE and Quantized Activation (QReLU) remains nearly consistent across all layers before and after retraining. Therefore, we propose to initially select the suppression method for each layer based on the standard-trained model, then proceed with suppression training using a mixed strategy of ELSE and another spatial method (e.g., STAR [64]). To configure the layerwise method for our mixed suppression Mix(ELSE, ϕ), we follow Equation (9) but set a conversion constraint γ on the layerwise compute reduction from ϕ to ELSE:

$$\begin{aligned} Objective: & \min_{\alpha} \sum_{l=0}^{L} (\alpha_l m_l^{ELSE} + (1-\alpha_l) m_l^{\phi}) \\ Constraints: & m_l^{ELSE} / m_l^{\phi} \geq \gamma \end{aligned} \tag{10}$$

5 Experiments and Results

5.1 Experimental Settings

Datasets: We utilize two video datasets MPII [3] and UA-DETRAC [56] as testbeds to evaluate our method on Pose Estimation and Object Detection.
Network Architectures: We evaluate our suppression method ELSE and its mixed variants across several DNN architectures: ResNets (RN18, RN50) [27], MobileNets (MNV1 [31], MNV2 [48]) and EfficientNetLiteB0 (EN-Lite) [51]. These architectures encompass diverse DNN building blocks, including residual blocks, separable convolution blockss, inverted residual blocks, and NAS searched blocks, representing a comprehensive spectrum of DNN designs.

Evaluation Protocol: We take the accuracy, million events (MEvt), billion event-triggered multiply–accumulates (GMAC), event reduction ratio (E_{rr}), compute reduction ratio (M_{rr}) and state-memory-consumption (state mem.) as the

evaluation protocol metrics for our proposed method. In this paper, the reduction in MEvt/GMAC is consistently compared between static and dynamic measures. In Table 1, Table 2 and Table 3, **we highlight the lowest MEvt and GMAC for the suppressed variants meeting hardware memory constraints.**

Compared methods: The event suppression methods can fall into two categories: (1) Spatial methods minimizing the non-zero activations, including L_1 [21], Square-Hoyer [35], STAR [64]; (2) Temporal methods suppressing the non-zero activation changes in time domain, such as EvNet [16], DeltaCNN [45], DAL [61] and CATS [62]. Note that all the temporal methods in this paper maintain the same accuracy in both spatial and temporal execution. **Implementation Details** can be found in Appendix B.

5.2 Object Detection

Table 1 shows the event suppression results for MNV1-SSD [37] and MNV2-SSDLite [48] on UA-DETRAC traffic dataset. The first notable observation is that ELSE achieves MAC reductions of 6.49× and 3.14× for these two detectors, respectively, surpassing the SOTA spatial method STAR by an average of 37% in computation savings. Another salient trend is the significant superiority of temporal methods over spatial ones in suppression due to the high temporal correlation in UA-DETRAC. Although CATS leads across four key performance metrics (mAP@0.5, MEvt, GMAC, Mem.) among temporal methods, and consumes only half the state memory compared to EvNet and DeltaCNN, its substantial state memory footprint still prevents the suppressed models from fitting into the on-chip memory of commercial event-driven processors [12,13,43]. However, our Mix(ELSE, CATS†) can inherit the achieved low computations from CATS while consuming 2.17× less state memory than CATS. Overall, the MAC savings achieved by Mix(ELSE, CATS†) can reach up to 10.32× and 3.99× on MNV1-SSD and MNV2-SSDLite, respectively, without compromising accuracy.

5.3 Human Body Pose Estimation

In addition to Object Detection, we evaluate our method across three widely-used backbones for SimpleBaseline [58] pose estimator and compare ELSE with activation suppression and temporal suppression in Table 2 and Table 3. First, we conjugate that ELSE achieves 2.43 $\sim$ 5.75× MAC reduction from light to heavy weight backbones, showing its superior suppression effect for weight-redundant networks. Secondly, our spatial mixed approach Mix(ELSE, STAR) can further boost the MAC reduction to 2.53 $\sim$ 8.83×, significantly surpassing the compared spatial SOTAs. Furthermore, Table 3 illustrates that our Mix(ELSE, DAL†) and Mix(ELSE, CATS†) consume the minimal state memory while achieving comparable event and MAC suppression as their full-temporal counterparts. For instance, Mix(ELSE, CATS†) with EN-Lite backbone reduces state memory by 2.84× with only a 3.53% increase in computation compared to CATS. Overall, our spatio-temporal mixed approach Mix(ELSE, CATS†) can achieve 2.68 $\sim$ 6.76× in computation reduction, with substantial state memory savings of 2.78 $\sim$

Table 1: A performance comparison with SOTA spatial and temporal methods for MNV1-SSD and MNV2-SSDLite on UA-DETRAC. We evaluate model accuracy using mAP@0.5(%) [18] and quantify the state memory footprint in 16 bits. Methods marked with the symbol † stand for temporal approach. ✓ denotes compliance with on-chip memory constraints of commercial event-driven processors [12, 13, 43].

Backbone	Method	mAP@0.5 ↑	MEvt ↓ (E_{rr} ↑)	GMAC ↓ (M_{rr} ↑)	Mem. ↓	fit
MNV1-SSD	ReLU	48.13%	6.03 (1.67x)	0.587 (2.08x)	1.28 MB	✓
	STAR [64]	48.02%	4.92 (2.05x)	0.333 (3.67x)	1.28 MB	✓
	ELSE	47.94%	3.20 (3.16x)	0.188 (6.49x)	1.86 MB	✓
	EvNet† [16]	48.01%	2.56 (3.95x)	0.105 (11.61x)	40.40 MB	
	DeltaCNN† [45]	48.01%	1.73 (5.83x)	0.151 (8.10x)	40.40 MB	
	CATS† [62]	47.95%	2.28 (4.42x)	0.090 (13.53x)	20.20 MB	
	Mix(ELSE, CATS†)	47.95%	**2.81 (3.59x)**	**0.118 (10.32x)**	9.22 MB	✓
MNV2-SSDLite	ReLU	48.72%	8.62 (1.81x)	0.454 (1.48x)	1.83 MB	✓
	STAR [64]	49.02%	7.45 (2.09x)	0.306 (1.48x)	1.83 MB	✓
	ELSE	49.04%	4.80 (3.25x)	0.213 (3.14x)	2.76 MB	✓
	EvNet† [16]	48.76%	4.35 (3.59x)	0.181 (3.70x)	62.38 MB	
	DeltaCNN† [45]	48.75%	3.23 (4.83x)	0.189 (3.55x)	62.38 MB	
	CATS† [62]	49.04%	3.56 (4.39x)	0.149 (4.51x)	31.19 MB	
	Mix(ELSE, CATS†)	49.04%	**4.20 (3.71x)**	**0.168 (3.99x)**	14.56 MB	✓

3.90×, highlighting an excellent computation/memory trade-off. Additionally, we notice that ELSE and its mixed variants occasionally achieve higher accuracy than the standard-trained model (ReLU) despite reduced computation. This may be due to the regularization effect of sparsity-inducing norms.

Table 2: A performance comparison with SOTA activation suppression methods on SimpleBaseline/MPII with various backbones. We evaluate model performance using PCK@0.5(%) and quantify the state memory footprint in 16 bits.

Backbone	Method	PCK@0.5 ↑	MEvt ↓ (E_{rr} ↑)	GMAC ↓ (M_{rr} ↑)	State Mem. ↓
MNV1 [31]	ReLU	84.06%	5.58 (1.97x)	0.638 (1.85x)	1.15 MB
	L_1 [21]	83.77%	4.68 (2.35x)	0.450 (2.62x)	1.15 MB
	Square-Hoyer [35]	83.78%	4.17 (2.63x)	0.385 (3.05x)	1.15 MB
	STAR [64]	84.05%	4.48 (2.45x)	0.400 (2.95x)	1.15 MB
	ELSE	84.10%	3.36 (3.27x)	0.216 (5.43x)	1.73 MB
	Mix(ELSE, STAR)	84.06%	**3.11 (3.53x)**	**0.177 (6.65x)**	1.32 MB
EN-Lite [52]	ReLU	86.17%	7.05 (1.81x)	0.549 (1.57x)	1.76 MB
	L_1 [21]	85.91%	6.38 (2.00x)	0.416 (2.07x)	1.76 MB
	Square-Hoyer [35]	85.93%	6.74 (1.89x)	0.509 (1.69x)	1.76 MB
	STAR [64]	86.12%	5.89 (2.16x)	0.360 (2.39x)	1.76 MB
	ELSE	86.07%	5.40 (2.36x)	0.355 (2.43x)	2.49 MB
	Mix(ELSE, STAR)	86.10%	**5.14 (2.48x)**	**0.341 (2.53x)**	1.96 MB
RN50 [27]	ReLU	87.70%	7.76 (2.17x)	3.502 (2.74x)	1.59 MB
	L_1 [21]	87.42%	3.06 (5.50x)	1.466 (6.54x)	1.59 MB
	Square-Hoyer [35]	87.40%	3.41 (4.93x)	1.693 (5.66x)	1.59 MB
	STAR [64]	87.44%	2.10 (7.99x)	1.431 (6.70x)	1.59 MB
	ELSE	87.46%	3.40 (4.95x)	1.667 (5.75x)	2.63 MB
	Mix(ELSE, STAR)	87.63%	**1.90 (8.83x)**	**1.085 (8.83x)**	1.73 MB

5.4 Ablation Study

Effect of suppression components in ELSE. Table 4 illustrates the suppression effect of each component in ELSE and its temporal variant Mix(ELSE,

Table 3: A performance comparison with SOTA temporal suppression methods on SimpleBaseline/MPII with various backbones. We evaluate model performance using PCK@0.5(%) and quantify state memory footprint in 16 bits. ✓ denotes compliance with on-chip memory constraints of commercial event-driven processors [12, 13, 43].

Backbone	Method	PCK@0.5 ↑	MEvt ↓ (E_{rr} ↑)	GMAC ↓ (M_{rr} ↑)	State Mem. ↓	fit
MNV1 [31]	EvNet† [16]	83.48%	3.68 (2.99x)	0.306 (3.84x)	43.64 MB	
	DeltaCNN† [45]	83.54%	4.06 (2.71x)	0.435 (2.70x)	43.64 MB	
	DAL† [61]	84.09%	3.38 (3.24x)	0.310 (3.79x)	21.82 MB	
	DAL-Select† [61]	84.09%	5.09 (2.16x)	0.529 (2.22x)	**7.65 MB**	✓
	CATS† [62]	84.10%	2.96 (3.71x)	0.164 (7.18x)	21.82 MB	
	Mix(ELSE, DAL†)	84.09%	3.66 (3.00x)	0.325 (3.62x)	**7.84 MB**	✓
	Mix(ELSE, CATS†)	84.10%	**3.27 (3.35x)**	**0.174 (6.76x)**	**7.84 MB**	✓
EN-Lite [52]	EvNet† [16]	86.13%	4.66 (2.73x)	0.310 (2.77x)	50.92 MB	
	DeltaCNN† [45]	86.16%	4.53 (2.81x)	0.347 (2.48x)	50.92 MB	
	DAL† [61]	85.80%	4.99 (2.55x)	0.350 (2.46x)	25.46 MB	
	DAL-Select† [61]	85.80%	6.57 (1.94x)	0.473 (1.82x)	**8.75 MB**	✓
	CATS† [62]	86.07%	4.75 (2.67x)	0.311 (2.77x)	25.46 MB	
	Mix(ELSE, DAL†)	85.80%	5.37 (2.37x)	0.364 (2.37x)	**8.97 MB**	✓
	Mix(ELSE, CATS†)	86.07%	**5.18 (2.46x)**	**0.322 (2.68x)**	**8.97 MB**	✓
RN50 [27]	EvNet† [16]	87.35%	2.48 (6.78x)	1.716 (5.59x)	67.24 MB	
	DeltaCNN† [45]	87.48%	2.83 (5.94x)	1.323 (7.24x)	67.24 MB	
	DAL† [61]	87.55%	4.57 (3.68x)	2.449 (3.91x)	33.62 MB	
	DAL-Select† [61]	87.55%	6.35 (2.65x)	3.152 (3.04x)	**8.08 MB**	✓
	CATS† [62]	87.46%	2.81 (5.98x)	1.516 (6.32x)	33.62 MB	
	Mix(ELSE, DAL†)	87.55%	5.44 (3.09x)	2.362 (4.06x)	**8.61 MB**	✓
	Mix(ELSE, CATS†)	87.46%	**3.15 (5.34x)**	**1.592 (6.02x)**	**8.61 MB**	✓

$\phi^\dagger$). We quantify the relative event and MAC improvement of each suppression component by introducing them sequentially, while monitoring the state memory consumption. One notable observation is that the highest decrease in events and MACs happens in two key components: Δ-Σ modulation and training with sparsity-inducing penalties. Another noteworthy trend is that Mix(ELSE, CATS†) induces an additional substantial reduction in MAC due to the high temporal correlation in dataset, resulting in the minimal computations.

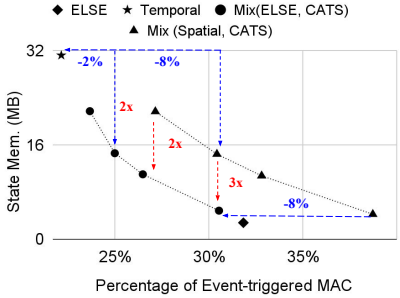
Effect of ELSE in spatio-temporal mixed suppression. Figure 7a compares our spatio-temporal mixed approach Mix(ELSE, $\phi^\dagger$) to the existing mixed approach [61] with respect to state memory/computation trade-offs. The key difference lies in the replacement of memory-heavy temporal layers performed by ELSE or spatially-executed CATS [62]. From left to right, both curves exhibit a significant reduction in memory usage initially, with this decrease gradually leveling off as MAC increases. A comparison of two curves reveals that our proposed approach Mix(ELSE, $\phi^\dagger$) in Equation (9) consistently surpasses the reference method by saving $2 \sim 3\times$ state memory cost at various computation degrees.

Effect of network architecture in event suppression. Figure 7b illustrates the normalized accuracy/computation curves of ELSE on various network backbones in SimpleBaseline [58], showing that heavy-weight networks (RN, EN-EM, MNV1) display better event suppression capacity than the light-weight ones (GN130, EN-Lite). We attribute this to the fact that heavy networks have more

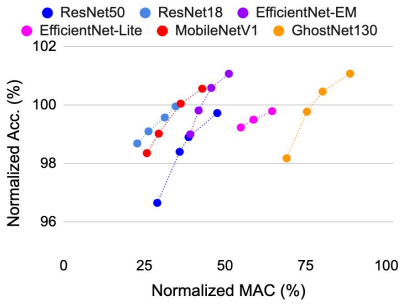
Table 4: Ablation Study of ELSE components.

Setting	SSD-MNV1/UA-DETRAC					
	Acc change	MEvt	Evt Imp	GMAC	MAC Imp	State Mem.
Conventional	-	10.10	-	1.22	-	1.28 MB
+ ReLU	+0.00%	6.03	40% ↓	0.59	52% ↓	1.28 MB
ELSE						
+ Δ-Σ	+0.00%	4.55	25% ↓	0.46	21% ↓	1.86 MB
+ quantization	+0.33%	3.95	13% ↓	0.34	27% ↓	1.86 MB
+ evt-aware penalty	-0.62%	3.28	17% ↓	0.31	10% ↓	1.86 MB
+ mac-aware penalty	-0.42%	3.33	-2% ↓	0.23	24% ↓	1.86 MB
+ line-based penalty	-0.39%	3.20	4% ↓	0.19	19% ↓	1.86 MB
+ mix ϕ: CATS [62]	-0.37%	2.82	12% ↓	0.12	37% ↓	9.22 MB

removable computations due to their weight redundancy. However, as shown in Table 1, the compact network (MNV2-SSDLite) do not necessarily achieve lower MACs after event suppression compared to the redundant one (MNV1-SSD). Given the importance of event-triggered MACs and memory for power and latency savings on event-driven processors, these findings suggest us to integrate sparsity into future DNN architecture design [8,57], such as sparsity-aware NAS.



(a) Comparison of memory and computation aspects between our Mix(ELSE, $\phi^{\dagger}$) and the existing DAL-Select† [61] when integrated with the temporal method CATS [62].



(b) Normalized accuracy/computation curves of our ELSE method across different DNN backbones for SimpleBaseline [58] pose estimator on MPII dataset.

Fig. 7: Effect of ELSE in spatio-temporal mixed suppression (a) and Effect of network architecture in event suppression (b).

6 Conclusion

We propose ELSE, a novel event suppression training method aimed at minimizing computations and memory accesses in event-driven DNN processing. Our primary contribution involves implementing Δ-Σ modulation between activation lines, promoting significant event sparsity with minimal state memory overhead. Additionally, ELSE can seamlessly integrate as a plug-and-play suppression module within DNN intermediate layers, compatible with other SOTA event suppression methods. Experimental results demonstrate that our spatial Mix(ELSE, ϕ) further enhances computation savings while our spatio-temporal Mix(ELSE, $\phi^{\dagger}$) reduces state memory footprint by over 2×, alleviating the challenge of temporal execution exceeding the resource constraints of real-world embedded platforms.

Acknowledgements

This work has been partially funded by the NimbleAI project. NimbleAI has received funding from the EU's Horizon Europe Research and Innovation programme (Grant Agreement 101070679), and by the UK Research and Innovation (UKRI) under the UK government's Horizon Europe funding guarantee (Grant Agreement 10039070).

References

1. Agustsson, E., Timofte, R.: Ntire 2017 challenge on single image super-resolution: Dataset and study. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 1122–1131 (2017). https://doi.org/10.1109/CVPRW.2017.150
2. Akopyan, F., Sawada, J., Cassidy, A.S., Alvarez-Icaza, R., Arthur, J.V., Merolla, P., Imam, N., Nakamura, Y., Datta, P., Nam, G.J., Taba, B., Beakes, M.P., Brezzo, B., Kuang, J.B., Manohar, R., Risk, W.P., Jackson, B.L., Modha, D.S.: Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems pp. 1537–1557 (2015)
3. Andriluka, M., Pishchulin, L., Gehler, P., Schiele, B.: 2d human pose estimation: New benchmark and state of the art analysis. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2014)
4. Bamberg, L., Pourtaherian, A., Waeijen, L., Chahar, A., Moreira, O.: Synapse compression for event-based convolutional-neural-network accelerators. IEEE Transactions on Parallel and Distributed Systems **34**(4), 1227–1240 (2023). https://doi.org/10.1109/TPDS.2023.3239517
5. Binas, J., Bengio, Y.: Low-memory convolutional neural networks through incremental depth-first processing. ArXiv (2018), https://api.semanticscholar.org/CorpusID:13755032
6. Boser, B., Wooley, B.: The design of sigma-delta modulation analog-to-digital converters. IEEE Journal of Solid-State Circuits **23**(6), 1298–1308 (1988). https://doi.org/10.1109/4.90025
7. BrainChip Ltd.: Akida neural processor system-on-chip (Feb 2021)
8. Cai, H., Zhu, L., Han, S.: ProxylessNAS: Direct neural architecture search on target task and hardware. In: International Conference on Learning Representations (2019), https://arxiv.org/pdf/1812.00332.pdf
9. Cavigelli, L., Benini, L.: Cbinfer: Exploiting frame-to-frame locality for faster convolutional network inference on video streams. IEEE Transactions on Circuits and Systems for Video Technology (5), 1451–1465 (2020). https://doi.org/10.1109/TCSVT.2019.2903421
10. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: ECCV (2018)
11. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proc. IEEE CVPR (2016)
12. Davies, M., Srinivasa, N., Lin, T.H., Chinya, G., Cao, Y., Choday, S.H., Dimou, G., Joshi, P., Imam, N., Jain, S., Liao, Y., Lin, C.K., Lines, A., Liu, R., Mathaikutty, D., McCoy, S., Paul, A., Tse, J., Venkataramanan, G., Weng, Y.H., Wild, A., Yang,

Y., Wang, H.: Loihi: A neuromorphic manycore processor with on-chip learning. IEEE Micro (1), 82–99 (2018)
13. Davies, M., Wild, A., Orchard, G., Sandamirskaya, Y., Guerra, G.A.F., Joshi, P., Plank, P., Risbud, S.R.: Advancing neuromorphic computing with loihi: A survey of results and outlook. Proceedings of the IEEE (5), 911–934 (2021). https://doi.org/10.1109/JPROC.2021.3067593
14. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Li, F.F.: Imagenet: a large-scale hierarchical image database. In: Proc. IEEE CVPR. pp. 248–255 (Jun 2009)
15. Deng, L., Wang, G., Li, G., Li, S., Liang, L., Zhu, M., Wu, Y., Yang, Z., Zou, Z., Pei, J., Wu, Z., Hu, X., Ding, Y., He, W., Xie, Y., Shi, L.: Tianjic: A unified and scalable chip bridging spike-based and continuous neural computation. IEEE Journal of Solid-State Circuits (8), 2228–2246 (2020). https://doi.org/10.1109/JSSC.2020.2970709
16. Dutson, M., Li, Y., Gupta, M.: Event neural networks. In: Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI. p. 276–293. Springer-Verlag, Berlin, Heidelberg (2022), https://doi.org/10.1007/978-3-031-20083-0_17
17. Elfwing, S., Uchibe, E., Doya, K.: Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. CoRR (2017)
18. Everingham, M., Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. Int. J. Comput. Vision **88**(2), 303–338 (jun 2010), https://doi.org/10.1007/s11263-009-0275-4
19. Furber, S.B., Galluppi, F., Temple, S., Plana, L.A.: The spinnaker project. Proceedings of the IEEE pp. 652–665 (2014)
20. Ge, Z., Liu, S., Wang, F., Li, Z., Sun, J.: Yolox: Exceeding yolo series in 2021. ArXiv **abs/2107.08430** (2021), https://api.semanticscholar.org/CorpusID:236088010
21. Georgiadis, G.: Accelerating convolutional neural networks via activation map compression. pp. 7078–7088 (06 2019). https://doi.org/10.1109/CVPR.2019.00725
22. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks (01 2010)
23. Goetschalckx, K., Wu, F., Verhelst, M.: Depfin: A 12-nm depth-first, high-resolution cnn processor for io-efficient inference. IEEE Journal of Solid-State Circuits (5), 1425–1435 (2023). https://doi.org/10.1109/JSSC.2022.3210591
24. Gupta, S., Akin, B.: Accelerator-aware neural network design using automl. ArXiv (2020), https://api.semanticscholar.org/CorpusID:212628608
25. Habibian, A., Abati, D., Cohen, T., Bejnordi, B.E.: Skip-convolutions for efficient video processing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2021)
26. Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., Xu, C.: Ghostnet: More features from cheap operations. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1577–1586 (2020). https://doi.org/10.1109/CVPR42600.2020.00165
27. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90
28. Höppner, S., Yan, Y., Dixius, A., Scholze, S., Partzsch, J., Stolba, M., Kelber, F., Vogginger, B., Neumärker, F., Ellguth, G., Hartmann, S., Schiefer, S., Hocker, T., Walter, D., Liu, G., Garside, J.D., Furber, S.B., Mayr, C.: The spinnaker 2 processing element architecture for hybrid digital neuromorphic computing. ArXiv (2021)

29. Horowitz, M.: 1.1 computing's energy problem (and what we can do about it). In: 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC). pp. 10–14 (2014). https://doi.org/10.1109/ISSCC.2014.6757323
30. Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y., Pang, R., Adam, H., Le, Q.: Searching for mobilenetv3. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 1314–1324 (2019). https://doi.org/10.1109/ICCV.2019.00140
31. Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications (04 2017)
32. Ignatov, A., Timofte, R., et al.: Pirm challenge on perceptual image enhancement on smartphones: report. In: European Conference on Computer Vision (ECCV) Workshops (January 2019)
33. Khoei, M.A., Yousefzadeh, A., Pourtaherian, A., Moreira, O., Tapson, J.C.: Sparnet: Sparse asynchronous neural network execution for energy efficient inference. 2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS) pp. 256–260 (2020), https://api.semanticscholar.org/CorpusID:216105549
34. Kurniawan, A.: Introduction to nvidia jetson nano pp. 1–6 (01 2021). https://doi.org/10.1007/978-1-4842-6452-2_1
35. Kurtz, M., Kopinsky, J., Gelashvili, R., Matveev, A., Carr, J., Goin, M., Leiserson, W., Moore, S., Nell, B., Shavit, N., Alistarh, D.: Inducing and exploiting activation sparsity for fast neural network inference. In: Proceedings of the 37th International Conference on Machine Learning. ICML'20, JMLR.org (2020)
36. Lim, B., Son, S., Kim, H., Nah, S., Lee, K.M.: Enhanced deep residual networks for single image super-resolution. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 1132–1140 (2017). https://doi.org/10.1109/CVPRW.2017.151
37. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: ECCV (1). Lecture Notes in Computer Science, vol. 9905, pp. 21–37. Springer (2016), http://dblp.uni-trier.de/db/conf/eccv/eccv2016-1.html#LiuAESRFB16
38. Lv, M., Xu, E.: Efficient dnn execution on intermittently-powered iot devices with depth-first inference. IEEE Access pp. 101999–102008 (2022), https://api.semanticscholar.org/CorpusID:252035232
39. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: in ICML Workshop on Deep Learning for Audio, Speech and Language Processing (2013)
40. Mei, L., Goetschalckx, K., Symons, A., Verhelst, M.: Defines: Enabling fast exploration of the depth-first scheduling space for dnn accelerators through analytical modeling. In: 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA). pp. 570–583. IEEE Computer Society, Los Alamitos, CA, USA (mar 2023). https://doi.org/10.1109/HPCA56546.2023.10071098
41. Moradi, S., Qiao, N., Stefanini, F., Indiveri, G.: A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs). IEEE Trans. Biomed. Circuits Syst. pp. 106–122 (Aug 2017)
42. Moreira, O., Yousefzadeh, A., Chersi, F., Kapoor, A., Zwartenkot, R.J., Qiao, P., Cinserin, G., Khoei, M., Lindwer, M., Tapson, J.: Neuronflow: A hybrid neuromorphic – dataflow processor architecture for ai workloads. In: 2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS). pp. 01–05 (2020). https://doi.org/10.1109/AICAS48895.2020.9073999

43. Moreira, O., Yousefzadeh, A., Chersi, F., Cinserin, G., Zwartenkot, R.J., Kapoor, A., Qiao, P., Kievits, P., Khoei, M.A., Rouillard, L., Ferouge, A., Tapson, J.C., Visweswara, A.: Neuronflow: a neuromorphic processor architecture for live AI applications. In: Proc. DATE. pp. 840–845 (2020)
44. O'Connor, P., Welling, M.: Sigma delta quantized networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net (2017), https://openreview.net/forum?id=HkNRsU5ge
45. Parger, M., Tang, C., Twigg, C.D., Keskin, C., Wang, R., Steinberger, M.: Deltacnn: End-to-end cnn inference of sparse frame differences in videos. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12487–12496 (2022). https://doi.org/10.1109/CVPR52688.2022.01217
46. Ramachandran, P., Zoph, B., Le, Q.V.: Searching for activation functions. ArXiv (2018)
47. Sabet, A., Hare, J.S., Al-Hashimi, B.M., Merrett, G.V.: Similarity-aware cnn for efficient video recognition at the edge. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems pp. 4901–4914 (2022), https://api.semanticscholar.org/CorpusID:245371897
48. Sandler, M., Howard, A.G., Zhu, M., Zhmoginov, A., Chen, L.: Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. CoRR (2018)
49. Stuijt, J., Sifalakis, M., Yousefzadeh, A., Corradi, F.: μbrain: An event-driven and fully synthesizable architecture for spiking neural networks. Front Neurosci. pp. 106–122 (May 2021)
50. Sze, V., hsin Chen, Y., Yang, T.J., Emer, J.S.: Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE pp. 2295–2329 (2017), https://api.semanticscholar.org/CorpusID:3273340
51. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. pp. 6105–6114. Proceedings of Machine Learning Research, PMLR (09–15 Jun 2019)
52. Tan, M., Le, Q.V.: (04 2020)
53. Tang, G., Safa, A., Shidqi, K., Detterer, P., Traferro, S., Konijnenburg, M., Sifalakis, M., van Schaik, G.J., Yousefzadeh, A.: Open the box of digital neuromorphic processor: Towards effective algorithm-hardware co-design. In: 2023 IEEE International Symposium on Circuits and Systems (ISCAS). pp. 1–5 (2023). https://doi.org/10.1109/ISCAS46773.2023.10181505
54. Waeijen, L., Sioutas, S., Peemen, M., Lindwer, M., Corporaal, H.: Convfusion: A model for layer fusion in convolutional neural networks. IEEE Access pp. 168245–168267 (2021). https://doi.org/10.1109/ACCESS.2021.3134930
55. Ward-Foxton, S.: GrAI Matter research gives rise to AI processor for the edge (May 2022), https://www.forbes.com/sites/karlfreund/2022/05/27/grai-matter-labs-brain-inspired-ai-for-the-edge/
56. Wen, L., Du, D., Cai, Z., Lei, Z., Chang, M.C., Qi, H., Lim, J., Yang, M.H., Lyu, S.: Ua-detrac: A new benchmark and protocol for multi-object detection and tracking. Comput. Vis. Image Underst. (apr 2020). https://doi.org/10.1016/j.cviu.2020.102907
57. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In: IEEE Conference on Computer Vision and Pattern

Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 10734–10742. Computer Vision Foundation / IEEE (2019). https://doi.org/10.1109/CVPR.2019.01099

58. Xiao, B., Wu, H., Wei, Y.: Simple baselines for human pose estimation and tracking. In: European Conference on Computer Vision (ECCV) (2018)
59. Xu, Y., Shidqi, K., van Schaik, G.J., Bilgic, R., Dobrita, A., Wang, S., Meijer, R., Nembhani, P., Arjmand, C., Martinello, P., Gebregiorgis, A., Hamdioui, S., Detterer, P., Traferro, S., Konijnenburg, M., Vadivel, K., Sifalakis, M., Tang, G., Yousefzadeh, A.: Optimizing event-based neural networks on digital neuromorphic architecture: a comprehensive design space exploration. Frontiers in Neuroscience **18** (2024). https://doi.org/10.3389/fnins.2024.1335422
60. Yoon, Y.C.: Lif and simplified srm neurons encode signals into spikes via a form of asynchronous pulse sigma–delta modulation. IEEE Transactions on Neural Networks and Learning Systems **28**(5), 1192–1205 (2017). https://doi.org/10.1109/TNNLS.2016.2526029
61. Yousefzadeh, A., Sifalakis, M.: Delta activation layer exploits temporal sparsity for efficient embedded video processing. In: 2022 International Joint Conference on Neural Networks (IJCNN). pp. 01–10 (2022). https://doi.org/10.1109/IJCNN55064.2022.9892578
62. Zhu, Z., Pourtaherian, A., Waeijen, L., Akkaya, I.B., Bondarev, E., Moreira, O.: Cats: Combined activation and temporal suppression for efficient network inference. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 8166–8175 (January 2024)
63. Zhu, Z., Pourtaherian, A., Waeijen, L., Bamberg, L., Bondarev, E., Moreira, O.: Arts: An adaptive regularization training schedule for activation sparsity exploration. In: 2022 25th Euromicro Conference on Digital System Design (DSD). pp. 415–422 (2022). https://doi.org/10.1109/DSD57027.2022.00062
64. Zhu, Z., Pourtaherian, A., Waeijen, L., Bondarev, E., Moreira, O.: Star: Sparse thresholded activation under partial-regularization for activation sparsity exploration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. pp. 4554–4563 (June 2023)