A. Activation Reduction vs. Suppression

Event-driven platforms [13, 46, 65] facilitate efficient DNN inference through two key properties: near-memory computing and event-driven processing. Near-memory computing reduces data movement costs by positioning memory in close proximity to logic units, while event-driven processing exploits activation sparsity by processing elements only when events (i.e., non-zero activations) occur. Both properties can significantly minimize latency and energy consumption during DNN inference.

Previous studies [18, 49, 50, 70] have shown significant progress in reducing computations through temporal suppression. However, this approach necessitates storing the intermediate features (i.e., neuron states) for each layer at time t-1 to generate inter-frame sparse differences at time t. As shown in Fig. 2, the state memory usage emerges as the primary bottleneck in mobile DNN architectures. This hinders the on-chip deployment of their temporal models, negating potential energy savings despite their significant advantages in computation reduction, as illustrated in Fig. 1.

To prevent the induction and accumulation of errors in long-term temporal Δ-Σ processing, MEET reduces the number of neuron states by trading fewer activations to more weights, while maintaining the linearity of Δ-Σ modulation (see Eq. (1)), referred to as **activation reduction**. To minimize dynamic computations, MEET suppresses non-zero values in the activation maps by leveraging temporal redundancy in videos, thereby improving event-driven processing, referred to as **activation suppression**.

More precisely, activation reduction is a modification of the network architecture, which removes feature maps or layers from the original network. It is similar to structural pruning [9, 43], as both aim to reduce activations. However, structural pruning can cause accuracy loss as it simultaneously reduces activation count, weight count, and computational load. In contrast, the A $\rightarrow$ W tradeoff in MEET reduces the activation count while increasing the weight count and computational load, thus maintaining accuracy.

On the other hand, activation suppression is an optimization method that preserves the original network architecture while dynamically reducing the number of non-zero activations during inference, based on the input data. On event-driven platforms, only non-zero activations trigger memory accesses and computations, resulting in substantial savings in latency and energy consumption [70, 72].

In summary, activation reduction minimizes the activation count without directly saving computations, whereas activation suppression saves computations without reducing the activation count. Prior research [7, 18, 36, 48–50, 70] have primarily focused on activation suppression to leverage the sparsity-aware properties of event-driven platforms, often overlooking the memory bottleneck associated with neuron states.

B. Temporal Sparsity and Network Design

As shown in Fig. 11, ENLite2 exhibits low static computations, while MEET-Full incurs heavy static computations. In an ideal scenario where there are no changes between video frames, no dynamic computation is triggered in either network, even if their static computations differ significantly. In a more realistic scenario, where some changes occur between frames (e.g., no and minor camera motion), dynamic computations increase in both networks, but the gap between their dynamic computations remains negligible. Since many evaluated videos in the MPII [2] dataset involve only moderate camera motion, the average dynamic computations of the two networks remain largely similar across the entire dataset. However, in a more challenging scenario with significant camera motion (e.g., major camera motion), the dynamic compute gap increases from 0 to 9 M, still relatively small compared to the total dynamic cycles. MEET still achieves a 1.7× cycle improvement over ENLite2 (ReLU Sparsity). In the worst-case scenario, where random frames exhibit zero temporal correlation, the compute gap increases, making temporal suppression unadvisable (see Appendix E). Thus, we believe that increasing static computations can be a viable strategy if it enhances other critical metrics (e.g., model accuracy, memory footprint), as the increased computations are likely offset by temporal sparsity.

Figure 11. EfficientNetLite2 (light) and MEET-Full (heavy) are evaluated across videos with different temporal correlations in MPII dataset.

C. Mix Spatial-Temporal Execution

MEET incorporates a mixed network architecture design that considers both the impact of suppression methods and the memory constraints of real-world hardware. First, we observe that the deeper layers of the network have fewer states because of their smaller feature sizes, but exhibit a stronger temporal suppression effect. Secondly, as noted in [64, 69], fine-grained (e.g., layerwise) mixed spatial-temporal execution in DNNs is memory-intensive. This is because n+1 layer states are required for n consecutive temporal layers, meaning that fewer consecutive temporal layers result in increased state memory costs.

Therefore, we divide the network into stages based on the feature downscale factor λ, with each stage consisting of the same block type coupled with specific suppression methods, as shown in Fig. 12. As an example of MEET-Mix@16×, if $\lambda > 16$, we retain the Inverted Bottleneck Blocks (MB) and reduce dynamic computations through spatial suppression [72]. Otherwise, we apply CSM-NAS (search space: FuseV2 block) to minimize the state size and reduce dynamic computations through temporal suppression [18]. In the special case where λ equals 1, CSM-NAS is applied to the entire network, referred to as MEET-Full.

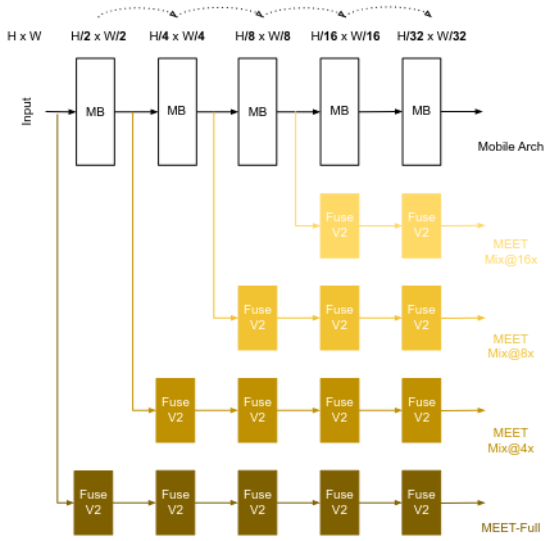


Figure 12. Overview of MEET-Full and MEET-Mix@λ×.

D. Activation → Weight Tradeoff in CSM-NAS

Our CSM-NAS enables a more fine-grained and efficient A → W tradeoff compared to the original FuseV2 MB block, with reduced manual efforts. In MEET, CSM-NAS incorporates multi-objective constraints within its search process. As shown in Fig. 13, under a given state memory budget, applying only a compute constraint produces candidates (purple) with minimal cycles, but leading to reduced accuracy. In contrast, applying only a memory constraint results in candidates (green) with increased redundant cycles. However, when both memory and compute constraints are incorporated, the resulting candidates (yellow) maintain high accuracy while minimizing computational load. Preserving low static cycles is essential, as temporal sparsity can substantially narrow the dynamic computation gap between light-weight and heavy-weight networks but cannot fully equalize dynamic computations across all networks.

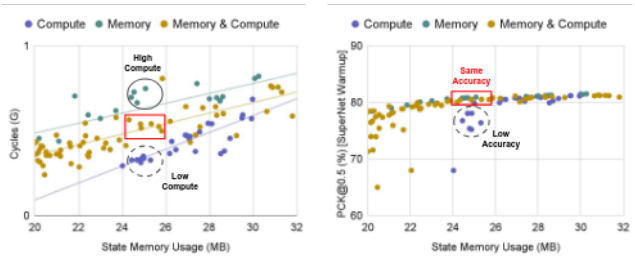


Figure 13. Effectiveness of the compute & memory constraint in CSM-NAS for compute and state memory efficiency at the same accuracy levels.

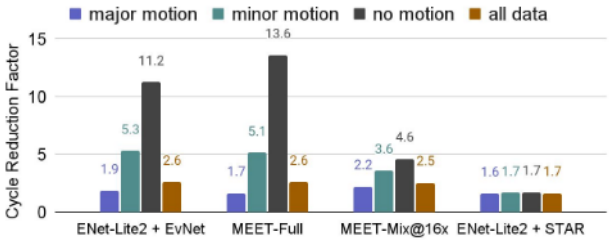


Figure 14. **Effect of camera motion on temporal, spatial and mix suppression.** Models trained and optimized with SimpleBaselines [61] (backbone: EfficientNet-Lite2 (ENet-Lite2) [55]) are evaluated on motion-classified MPII datasets [18]. Grouped by accuracy, we show the cycle reduction of each approach relative to the ReLU-Sparsified ENet-Lite2 across various camera motions.

E. Motion Effects on Event Suppression

E.1. Experimental setup

Datasets and Applications: We conduct experiments on human body pose estimation task using MPII [2] video dataset. To show the motion effects on event suppression, followed by previous work [18], we split the MPII [2] dataset into four motion classes: major motion, minor motion, no motion, and all data.

Network Architectures: We evaluate MEET by replacing the network backbone with CSM-NAS networks in SimpleBaselines [61]. We use EfficientNet-Lite2 (ENet-Lite2) [55] as a reference backbone, as it optimizes network efficiency in static computations, resulting in superior performance in dynamic computations compared to other widely-used networks [30, 53].

Methods: We employ STAR [72] and EvNet [18] for spatial and temporal suppression, respectively, as each represents the state-of-the-art (SOTA) in suppression performance within its respective category.

Evaluation Protocol: We follow established evaluation metrics from the main paper, including accuracy (PCK@0.5), dynamic cycles (Dyn Cycles), state memory usage (State Mem.), and total memory usage (Total Mem.) encompassing weights and states. We use the performance estimator, GrAIFlow [46], to measure the DNN inference cycles on an event-driven platform GrAI-VIP [59]. Tempo-

ral thresholds are derived from a subset of the training data and then evaluated across the entire validation set, with values fixed for different motion studies to maintain consistent model accuracy. (Additional implementation details can be found in Appendix G).

Performance Comparisons: We observe significant variation in event sparsity across network architectures: heavyweight networks (e.g., ResNet [30], VGG [53]) achieve greater cycle reduction compared to lightweight networks (e.g., MobileNets [32, 51], EfficientNets [55]), but with a higher number of absolute cycles, as shown in Tab. 3 and Tab. 5. To ensure a fair comparison, we normalize the results by setting the dynamic cycles of ReLU-Sparsified ENet-Lite2 as the baseline, presenting relative improvements across various networks and suppression methods .

E.2. Spatial vs. Temporal Suppression

As shown in Fig. 14, each approach shows cycle reduction relative to ReLU-Sparsified ENet-Lite2 under various camera motions, transitioning from temporal to spatial suppression (left to right). We first look into all data evaluation (brown) to compare different suppression methods. One salient trend is that temporal suppression (ENet-Lite2 + EvNet, MEET-Full) and mix suppression (MEET-Mix@16×) achieve 2.6× and 2.5× computation savings, significantly outperforming the 1.7× savings in spatial suppression (ENet-Lite2 + STAR). The second noticeable trend is that both MEET-Full and MEET-Mix@16× achieve a similar cycle reduction as ENet-Lite2 + EvNet, indicating that MEET can effectively maintain the computation efficiency while reducing neuron states.

E.3. Motion Effects

Fig. 14 compares cycle reductions across different camera motions for each approach. Spatial suppression (ENet-Lite2 + STAR) shows a consistent reduction across all camera motions, indicating a negligible impact from camera motion. In contrast, the cycle reduction for temporal suppression (ENet-Lite2 + EvNet, MEET-Full) varies significantly. For example, MEET-Full increases the cycle reduction from 1.7× to 13.6× as the camera motion decreases from major motion to no motion. More precisely, MEET-Full achieves cycle reduction similar to spatial suppression under major motion, but achieves 5.1× and 13.6× reductions under minor and no motion, respectively, significantly surpassing the 1.7× reduction in spatial suppression and demonstrating potential for ultra-efficient DNN inference.

In addition, MEET-Mix@16× demonstrates a superior cycle reduction of 2.2× in major motion, but achieves moderate reductions of 3.6× and 4.6× under minor and no motion, respectively. This is due to the use of spatial suppression in the initial network stages (λ >16), which is similar under major motion but worse under minor or no motion compared to temporal suppression, as shown in Fig. 15. However, Fig. 16 shows that computations are more intensive in the deeper stages than in the initial stages; therefore, MEET-Mix@16× can significantly benefit from decreased camera motion in the deeper stages, resulting in overall better cycle reduction compared to spatial suppression. Notably, MEET-Mix@16×, a mixed spatial-temporal suppression approach, consistently outperforms spatial suppression in cycle reduction and exhibits better robustness to motion effects compared to temporal suppression.

Lastly, MEET-Full consistently achieves cycle reductions similar to ENet-Lite2 + EvNet across various camera motions and network stages, as illustrated in Fig. 17, highlighting its capability to maintain the computation efficiency of ENet-Lite2 + EvNet under all conditions.

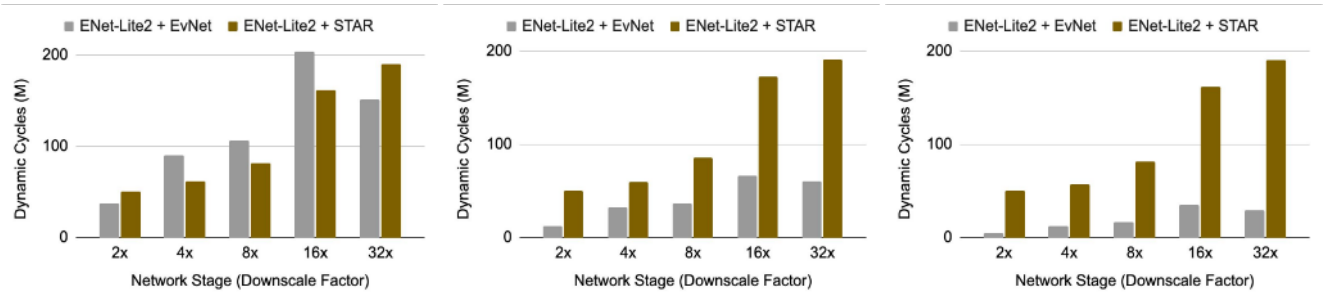
E.4. Hardware Deployment

In our study, we carry out the experiments on an event-driven platform GrAI-VIP [46, 59], which is equipped with 36 MB on-chip memory (SRAM). We assess the impact of different camera motions (see Fig. 14) while considering hardware memory constraints (see Tab. 4) to identify the optimal candidate in various scenarios.

Major camera motion: MEET-Mix@16× demonstrates a superior cycle reduction compared to all the others. It proves to be the optimal choice for such scenarios, achieving a 2.2× cycle reduction with a total memory cost of 12.8 MB, well within the limit of on-chip memory.

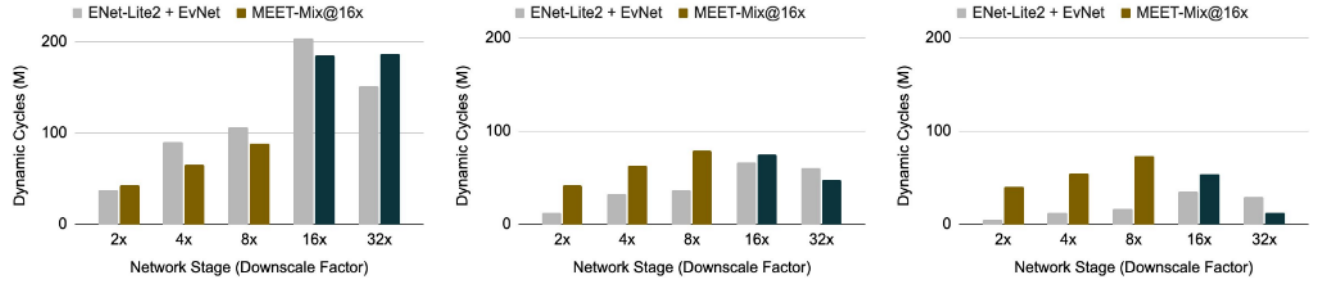
Minor or no camera motion: Temporal suppression significantly outperforms spatial suppression in cycle reduction. However, the SOTA temporal approach (ENet-Lite2 + EvNet) requires 66.2 MB of state memory usage, far exceeding the on-chip memory constraint. MEET-Full and MEET-Mix@16× address this issue by reducing state memory usage to 26.6 MB and 6.8 MB, respectively, enabling temporal networks to deploy within SRAM. Notably, MEET-Full achieves superior cycle reductions of 5.1× and 13.6× under minor and non-camera motion, respectively, preserving the computation efficiency of ENet-Lite2 + EvNet while surpassing the 3.6× and 4.6× reductions of MEET-Mix@16×. These advantages establish MEET-Full as the optimal candidate for such scenarios.

All camera motion types (all data): MEET-Mix@16× achieves a cycle reduction comparable to temporal suppression while significantly outperforming spatial suppression. With the benefit of lower memory usage, MEET-Mix@16× emerges as the optimal choice, delivering a 2.5× cycle reduction with a total memory cost of 12.8 MB. However, Fig. 16 shows that MEET-Mix@16× fails to match the cycles of ENet-Lite2 + EvNet in the initial network stages under both minor and no camera motion. We infer that major motion videos constitute a significant portion of the entire



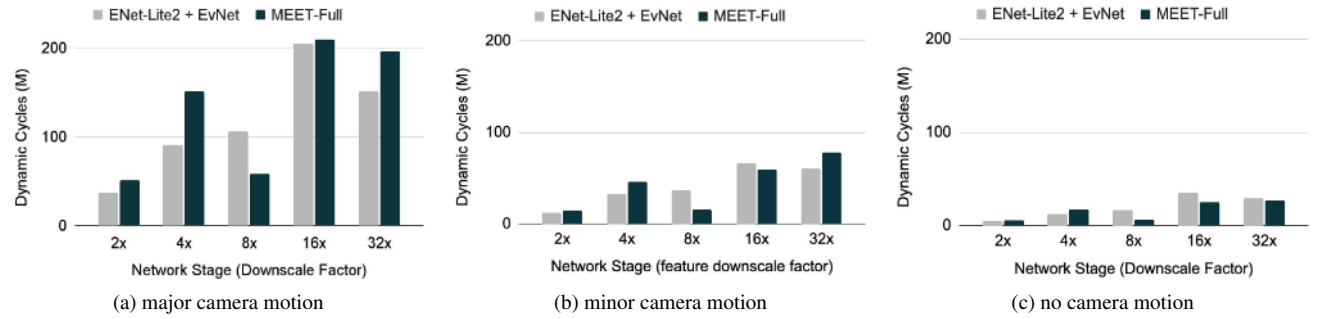
(a) major camera motion (b) minor camera motion (c) no camera motion

Figure 15. Dynamic cycle comparison between ENet-Lite2 + EvNet and ENet-Lite2 + STAR across network stages.



(a) major camera motion (b) minor camera motion (c) no camera motion

Figure 16. Dynamic cycle comparison between ENet-Lite2 + EvNet and MEET-Mix@16× across network stages.



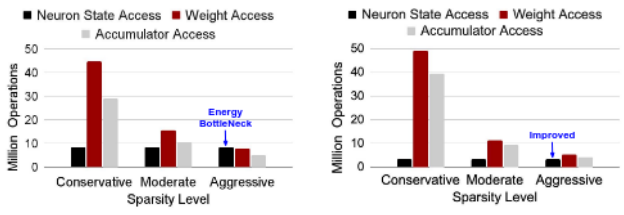
(a) major camera motion (b) minor camera motion (c) no camera motion

Figure 17. Dynamic cycle comparison between ENet-Lite2 + EvNet and MEET-Full across network stages.

validation dataset, which substantially diminishes the contribution of minor and no motion videos in cycle reduction when averaged across the entire dataset. Thus, identifying a reliable metric to assess the dynamic performance of temporal networks under different camera motions presents an intriguing direction for future research.

In summary, MEET-Full and MEET@16× perform as the optimal candidates across various camera motion scenarios, outperforming both SOTA temporal and spatial suppression when considering real-world hardware constraints. As a result, the deployment of our memory-efficient temporal Δ-Σ DNNs (MEET) harnesses the advantages of both near-memory computing and event-driven processing, which are essential for achieving low-power edge processing.

F. Energy Bottleneck in High Sparsity



(a) ENet-Lite2 + EvNet (b) MEET-Full

Figure 18. **Breakdown of memory access in (ENet-Lite2 + EvNet) versus MEET-Full for an event-driven platform.** MEET significantly reduces neuron state accesses, alleviating the energy bottleneck in high-sparsity event-driven processing.

As shown in Fig. 18, an interesting observation arises when we break down the number of memory access for (ENet-

Lite2 + EvNet) and MEET-Full. The number of neuron state access for the former approach is considerably higher than that of the latter one due to its much larger state size. Since the state access count remains constant regardless of sparsity, the high state access count of ENet-Lite2 becomes increasingly significant as network sparsity increases, ultimately becoming an energy bottleneck in highly sparse temporal models. However, MEET-Full substantially mitigates this issue by minimizing the state size.

G. Implementation Details

Standard Training: All standard models [30, 32, 51, 55] and CSM-NAS searched models are trained from scratch following an identical training recipe according to applications. All spatial and temporal (activation) suppression experiments are conducted on these standard-trained models.

Spatial Suppression: We employ STAR [72] for spatial suppression, a method that penalizes and thresholds low-magnitude activations during optimization training. The optimization training process is half the standard training epochs, with the learning rate reduced by a factor of 10 at every 1/3 epochs.

Temporal Suppression: We utilize EvNet [18] for temporal suppression, a method that truncates low-magnitude delta activations while mitigating truncation errors through long-term memory. A subset of the training data is used to finetune the optimal power-of-two values for layer-specific thresholds without training, achieving a balance between accuracy and dynamic cycle efficiency.

Mix Suppression: In MEET-Mix@$\lambda\times$ models, spatial suppression is applied exclusively to the activations of Inverted Bottleneck Blocks (MB), since MB trades fewer weights for more activations. In contrast, weight compression is applied solely to the weights of CSM-NAS Blocks (FuseV2), as FuseV2 trades fewer activations for more weights. In addition, temporal suppression, an optimization in an orthogonal direction to spatial optimization (e.g., spatial suppression, weight pruning, weight quantization, etc), is applied to CSM-NAS Blocks, causing negligible impact on weight compression. Importantly, block-specific optimization prevent excessive reduction in either weights or activations, thereby maintaining model accuracy.

Weight Compression: Following previous studies [9, 29, 57], we combine pruning [68] and quantization [38] to collaboratively compress the memory footprint of network weights. In particular, weight sparsity via pruning also aids dynamic computation reduction on event-driven processors, as each MAC is the inner product of an activation and a weight. However, we exclude it from the calculation of dynamic cycles to keep the focus of the study clear. Additionally, we apply quantization exclusively to weights, not activations, to mitigate accuracy drop. Low-bit weights not only reduce memory usage but also decrease the energy required for weight fetching, which is reflected in the active energy measures. While advanced weight compression methods like those in [9, 57] could yield additional savings, they are beyond the scope of this paper.

H. Broader Impact

MEET reduce the activation count, leading to reductions in both state memory and overall memory usage for temporal Δ-Σ DNNs, makcing them deployable within the on-chip memory of embedded event-driven platforms. In our study, we adopt EvNet [18] as our temporal firing mechanism due to its superior event suppression performance. However, MEET can benefit any temporal suppression method [26, 48–50, 64, 70] in memory reduction, regardless of their event firing mechanism, quantization or thresholding, etc.

Furthermore, Temporal Δ-Σ Networks share many similarities with Spiking Neural Networks (SNNs) [19], as both are inspired by the human brain and leverage sparsity in temporal domain for efficient computing. In SNNs, the number of neurons in a layer is often chosen to match the number of units in the feature map of the corresponding DNN layer to enable one-to-one mapping when converting a DNN to an SNN. Therefore, MEET can also benefit SNNs by reducing the neurons, thereby lowering the cost of off-chip data movement or alleviating the required neuron amount on event-driven platforms.