SPARSE-WAM: DECISION-AWARE SPARSE ATTENTION FOR EFFICIENT WORLD ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World Action Models (WAMs) jointly model future visual states and robot actions, but their large visual token sequences and iterative diffusion inference incur substantial computational costs. Existing sparse attention methods for video generation primarily preserve visual fidelity, which does not necessarily preserve the interactions required for action decisions. We propose Sparse-WAM, a decision-aware dynamic sparse attention framework for efficient WAM inference. Sparse-WAM calibrates attention budgets across heads and generation stages offline and adapts them through lightweight online sampling. It then uses action-to-video attention to identify action-relevant visual anchors and video-to-video dependencies to recover their supporting context, constructing sparse visual routes within the allocated budgets. To translate sparsity into practical acceleration, Sparse-WAM reuses stable budgets and calibration statistics across adjacent steps while updating sparse connections, and combines block-sparse execution with token reordering and cost-aware fallback. Experiments on LIBERO show that Sparse-WAM substantially reduces attention computation while preserving task performance, achieving an actual speedup of **XXX**×.

1 INTRODUCTION

Recent advances in video generation models have extended embodied policies beyond direct observation-to-action mapping toward joint modeling of world dynamics and robot actions. World Action Models (WAMs) typically build on large-scale pretrained DiTs to jointly generate future visual states and robot actions, thereby exploiting information about how the environment evolves to improve decision-making in complex tasks. However, the large number of visual tokens and multi-step DiT inference incur substantial computational costs, severely limiting the real-time deployment of WAMs. Attention among visual tokens is one of the primary computational bottlenecks.

Sparse attention reduces attention costs by computing only the important query–key interactions. Recent video generation methods have begun to exploit the spatiotemporal redundancy of attention and the heterogeneity across heads to introduce sparsity: for example, SVOO allocates sparsity budgets to different layers and heads, while TRA dynamically adjusts the spatiotemporal receptive field according to each query's attention distribution. However, these methods primarily aim to preserve visual generation quality, whereas WAMs prioritize action prediction and closed-loop task performance. Consequently, sparsity patterns designed for video generation may retain visually salient information while discarding the interactions essential for action selection. In other words, generation-preserving sparsity does not imply decision-preserving sparsity.

Our analysis of attention distributions in WAMs yields three findings. First, visual tokens vary substantially in their value for decision-making: information related to manipulation targets, contact regions, and state changes is typically more important. Second, action-to-video (A→V) and video-to-video (V→V) attention provide complementary information: A→V reflects how the action attends to different visual information, whereas V→V characterizes the dependencies among visual tokens. Action attention can therefore guide the priority of visual computation, but it cannot directly replace the selection of sparse connections on the visual side. Third, the computational budgets required by different attention heads and generation stages differ significantly; their overall budgets and statistical characteristics exhibit local stability, but the specific sparse connections still vary dynamically with the current input.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

These characteristics pose three challenges for sparse attention in WAMs: how to dynamically allocate computational budgets across heads and generation stages with low overhead; how to translate the action-relevance cues in A→V attention into reliable V→V sparsity structures without disrupting the necessary visual context; and how to organize dynamic, irregular sparsity patterns into hardware-friendly computation so that the theoretical reduction in FLOPs translates into inference speedup.

To this end, we propose **Sparse-WAM**, a decision-aware dynamic sparse attention framework for WAMs. Sparse-WAM first establishes base budgets for different heads and generation stages through offline calibration, and applies lightweight adjustments to them using a small number of online samples. It then locates action-relevant visual anchors with A→V attention and supplements the context required to support these anchors with V→V attention, thereby generating sparse visual routes. Finally, it reuses stable budgets and calibration statistics across adjacent steps while dynamically updating the specific sparse connections, and achieves end-to-end acceleration through token reordering, block-sparse operators, and a cost-aware fallback mechanism. Sparse-WAM thus answers the three core questions of sparse computation in WAMs: how much to compute, what to compute, and how to compute efficiently.

2 BACKGROUND

2.1 WAM MODELS

Existing WAMs are typically built on large-scale visual generation models or pretrained DiTs, representing environment states with visual tokens and robot control signals with action tokens. Depending on how visual prediction and action generation are organized, existing methods can be broadly categorized into **joint models** and **cascade models**.

2.1.1 JOINT MODEL

Joint WAMs model visual and action tokens simultaneously within a unified Transformer/DiT, involving **video-to-video (V→V)**, **action-to-video (A→V)**, and other cross-modal information flows. This unified formulation can fully exploit the coupling between vision and action. However, when the number of visual tokens far exceeds that of action tokens, large-scale **V→V attention** often dominates the overall attention computation. Therefore, how to exploit the information provided by the action side to reduce redundant computation on the visual side is an important efficiency problem for joint WAMs.

2.1.2 CASCADE MODEL

Cascade WAMs decouple world-state modeling from action prediction into relatively independent stages: a preceding model first predicts or constructs a representation of environment evolution, and an action model then generates robot actions conditioned on the current observation and the predicted visual states. Its action generation still relies heavily on the large-scale intermediate representations produced by the visual model, so the computational cost of visual modeling itself does not disappear.

Whether joint or cascade, reducing redundant interactions among visual tokens remains an important direction for achieving real-time inference in WAMs.

2.2 SPARSE ATTENTION OPTIMIZATION

Standard attention explicitly computes a large number of query–key interactions, and its computational cost grows quadratically with the number of tokens. Depending on the granularity of sparsity, existing methods typically allocate computational budgets at different levels such as **layer, head, token, or block**, and combine local windows, Top-K selection, dynamic routing, or block-sparse kernels to enable efficient computation.

2.2.1 SPARSE ATTN FOR MULTI-MODAL GENERATION MODELS

In video and multi-modal generation models, visual attention often exhibits pronounced spatiotemporal locality, head heterogeneity, and query-level sparsity, and a large body of work attempts to

exploit these structures to reduce computational cost. For video generation models, the importance of a token is determined mainly by its contribution to visual content reconstruction and generation quality. In WAMs, however, the same visual information must additionally serve robot action decision-making. Consequently, **generation-preserving sparsity does not inherently guarantee decision-preserving sparsity**.

2.2.2 SPARSE ATTN FOR WAM MODELS

Sparse attention in WAMs faces a more complex optimization objective. On the one hand, the number of visual tokens far exceeds that of action tokens, making **V→V attention** the most worthwhile component to compress. On the other hand, the importance of visual information is also governed by robot manipulation targets and action decisions, which makes purely vision-side sparsity rules unable to directly characterize which computations are truly worth preserving. Moreover, the visual context required by different attention heads and different generation stages in a WAM is not uniform. Some heads rely mainly on local regions, whereas others need to maintain broader spatiotemporal interactions; these patterns may further vary dynamically with the current input and the generation process. As a result, a unified window or a fixed budget can hardly balance efficiency and decision performance simultaneously, while relying entirely on token-level online profiling may introduce substantial scheduling overhead.

These characteristics indicate that sparse attention for WAMs must jointly account for **action-relevant information, internal visual dependencies, and dynamic computational budgets**. How to generate hardware-friendly visual sparsity structures at sufficiently low additional overhead while preserving decision-critical information is the core problem that this paper investigates further.

3 ANALYSIS

To understand why sparse attention in video generation is difficult to transfer directly to WAMs, we analyze the attention behavior of WAMs from two perspectives: **computational budget** and **cross-modal information interaction**. The analysis shows that the sparse structure of a WAM simultaneously exhibits pronounced head/stage differences, a complementary relationship between action and visual information, and a tension between fine-grained sparsity and hardware efficiency. Together, these phenomena determine the design space of sparse attention for WAMs.

3.1 HEAD- AND STAGE-WISE SPARSITY HETEROGENEITY

We first analyze the dependence of different attention heads and generation stages on visual context. For each head, we gradually lower the retention budget of V→V attention and observe the change in the final action output. The results show that the sensitivity of different heads to sparsification differs markedly: some heads maintain stable decisions even at low budgets, whereas others degrade noticeably at the same sparsity ratio. Setting a uniform computational budget for all heads therefore causes both **redundant computation and the loss of critical dependencies**.

Furthermore, the budget requirement of the same head also changes with the generation stage. Different stages carry different visual interaction functions, so the degree of sparsity they can tolerate is not fixed. We observe, however, that between adjacent generation steps the **overall budget requirement and statistical characteristics exhibit relatively strong local stability**, whereas the positions of the specific high-response tokens change more frequently. This means that the budget does not need expensive token-level profiling at every step, but can be modeled in the coarser-grained Head $\times$ Stage space, with only lightweight adjustment for local changes.

This analysis reveals the first key tension: **a WAM needs a more fine-grained dynamic budget than uniform head-level sparsity, yet it cannot afford the additional cost of full token-level profiling.** This tension directly determines our choice to use Head $\times$ Stage as the budget modeling granularity, to move the main analysis offline, and to retain only lightweight online adjustment.

3.2 Complementarity between Action Guidance and Visual Dependency

We further study whether action information can be used directly to compress the large visual attention. A→V attention usually highlights the visual tokens related to the current manipulation target, contact regions, and state changes, and can therefore provide decision cues that are more direct than purely visual saliency.

However, when we take the high-response visual positions in A→V directly as the sparse region of V→V, clear information loss still occurs. The reason is that A→V describes **which visual states are relevant to the action decision**, whereas V→V additionally has to model the spatial and temporal context that supports these states. For example, a manipulation object that is highly attended by the action side still depends on its neighboring regions, historical states, and motion trajectory to form a stable visual representation.

A further comparison of the two attention distributions shows that A→V and V→V are clearly correlated on manipulation-related regions, but no strict one-to-one correspondence exists between them. The two are therefore better understood as a **complementary relationship**: A→V provides action-driven visual anchors, while V→V supplies the internal visual dependencies required by these anchors.

This phenomenon shows that action information can indeed substantially narrow the visual search space, but **cannot directly replace the internal structure of V→V**. The second core problem is therefore how to build a reliable bridge from action cues to the visual sparsity structure, so that the action-relevance prior and the visual-side dependency structure jointly determine the sparse connections.

3.3 From FLOPs Reduction to Real Speedup

Finally, we analyze whether fine-grained sparsity can directly bring actual inference speedup. Token-level sparsity can achieve a higher FLOPs reduction in theory, but the sparse connections it produces are usually highly irregular: the retained length and the accessed positions may differ across heads and across queries.

This irregularity introduces additional overhead for index generation, gather, sorting, data reordering, and kernel dispatch, while reducing the parallelism and memory-access contiguity of the GPU. Consequently, under the same theoretical sparsity, the actual latency reduction of a fine-grained irregular mask is often markedly lower than its FLOPs reduction; when the sparsity ratio is low or the access pattern is too scattered, the computation may even decrease while the actual inference speed does not improve.

At the same time, the local stability observed in Section 3.1 offers another clue: although the specific sparse connections keep changing, the budget and part of the statistical information can be reused across adjacent steps, thereby reducing repeated profiling and scheduling cost. In addition, organizing the fine-grained connections into more regular blocks also helps improve GPU utilization.

The last problem of sparse attention for WAMs is therefore not simply how to obtain a higher sparsity, but **how to turn a dynamic sparse structure into hardware-friendly actual computation, so that the theoretical FLOPs reduction ultimately materializes as an end-to-end latency reduction.**

4 Method

4.1 Action-Aware Head Profiling and Adaptive Budget Allocation

We choose **Head** $\times$ **Stage** as the granularity of budget allocation, moving the main analysis offline and retaining only lightweight online adjustment.

Offline, with the input, history, and sampling process held fixed, we impose candidate sparsity budgets on different heads and continue running the complete inference, measuring each head's decision sensitivity by the effect of sparsification on the final action output. Combining the statistical characteristics of A→V and V→V attention, we further distinguish heads with pronounced action

guidance, heads with a strong dependence on visual context, and heads that are sensitive to sparsity, and we establish a base budget for each generation stage separately.

For an execution unit $u = (l, h, s)$, budget selection is written as

$$\rho_u^* = \arg\min_{\rho \in \mathcal{B}} C(u, \rho), \qquad \text{s.t.} \quad \Delta_{\text{act}}(u, \rho) \le \epsilon, \tag{1}$$

where $\mathcal{B}$ is the discrete budget set, C denotes the actual computational cost, and Δ_{act} measures the change in action decisions. This objective takes decision preservation directly as its constraint rather than relying only on visual reconstruction error.

Online, we sample only a small number of visual queries to apply a bounded correction to the offline budgets: stable heads directly reuse their base budget; when the action-related error grows, the budget is raised; when the reliability of the action prior decreases, a more conservative configuration is adopted. All adjustments are restricted to the discrete budget buckets, and the profiling overhead is charged to the actual latency. M1 finally outputs the computational budget of each head at the current stage, answering "**how much to compute**" for the subsequent sparse routing.

4.2 Action-Guided Visual Dependency Bridging and Sparse Routing

In WAMs, A→V attention and V→V attention are correlated but not equivalent, so the Top-K positions of A→V cannot be used directly as the V→V sparse mask. Sparse-WAM treats A→V as an action prior and then uses the internal visual dependencies to complete the context, thereby bridging action semantics to the visual sparsity structure.

Concretely, for head h and visual token j, we first aggregate the attention of all action queries to that token to obtain the action-guided score $s_{h,j} = \text{Agg}_{a \in \mathcal{A}} \alpha_{a,j}^{(h)}$. Keeping only the action anchors can easily destroy the integrity of the visual context, so we further expand their support region according to the local affinity of V→V. For candidate visual blocks i, j, we estimate with block-level Q/K representations $C_{hij} = \frac{\bar{Q}_{hi} \bar{K}_{hj}^\top}{\sqrt{d}}$, and preferentially retain the temporal neighborhood and visual blocks that have high visual dependency on the action anchors. Finally, under the budget ρ_u given by M1, we generate the V→V sparse connections that are actually executed,

$$R_u = \text{Anchor}(A_u) \cup \text{Context}(A_u) \cup \text{Structural}, \tag{2}$$

and weight them according to the head types obtained from M1, selecting the visual tokens with the highest scores as **action anchors**. Since the number of A→V queries is far smaller than the number of visual queries, this step introduces only low additional overhead.

4.3 Cross-Step Reuse and Hardware-Efficient Sparse Execution

Dynamically produced token-level sparsity patterns are usually strongly irregular; executing them directly introduces additional profiling, gather, reordering, and kernel dispatch overhead, so that a reduction in theoretical FLOPs is difficult to translate into real speedup. Sparse-WAM therefore further optimizes the actual computation along three aspects: **cross-step reuse, block-level regularization, and cost-aware execution**.

Between adjacent generation steps, the **budget requirement and statistical characteristics change relatively slowly, whereas the specific positions of interest change faster**. We therefore treat the budget ρ_u, the head type c_u, and the calibration statistics θ_u as slow variables, and the sparse connections R_u as fast variables:

$$Z_u^{\text{slow}} = (\rho_u, c_u, \theta_u), \qquad Z_u^{\text{fast}} = R_u. \tag{3}$$

For adjacent steps $t-1$ and t, when the drift of the current statistical characteristics is below a threshold, the existing budget and calibration state are reused directly:

$$\text{Reuse}(u, t) = \mathbb{I}\left[D\left(\phi_u^t, \phi_u^{t-1}\right) < \tau_u\right], \tag{4}$$

where ϕ_u denotes the lightweight features formed by the sampled queries, the action anchors, and local visual statistics.

Second, to reduce the execution overhead caused by irregular memory access, we further regularize the token connections output by M2 into a block-level sparse structure. Let each visual block contain B tokens; the set of blocks actually executed is defined as $\mathcal{B}_u = \{\, b \mid R_u \cap b \neq \varnothing \,\}$.

We then bucket the heads according to the number of retained blocks and reorder the selected Key/Value blocks into contiguous storage, so that the execution shapes within the same computation bucket are similar. Sparse attention is computed only on the selected blocks:

$$O_u = \mathrm{Softmax}\left(\frac{Q_u K_{\mathcal{B}_u}^{\top}}{\sqrt{d}}\right) V_{\mathcal{B}_u}. \tag{5}$$

Finally, we explicitly charge all additional scheduling costs to the actual latency. For an execution unit u, define

$$T_u^{\text{sparse}} = T_u^{\text{route}} + T_u^{\text{reorder}} + T_u^{\text{kernel}}. \tag{6}$$

The sparse path is enabled only when $T_u^{\text{sparse}} < T_u^{\text{dense}}$; otherwise we fall back directly to dense attention. In this way, Sparse-WAM avoids the situation in which "FLOPs decrease but actual latency rises" at high budgets or when the sparse structure is overly scattered, and turns dynamic sparsity into genuine end-to-end inference acceleration through **cross-step reuse, block-level regularization, and cost-aware fallback**.

5 Experiments

We systematically evaluate Sparse-WAM from two aspects, **task performance and inference efficiency**: whether Sparse-WAM can preserve the closed-loop decision capability of a WAM while substantially reducing the attention computation cost, and how each of the three core modules affects the final performance–efficiency trade-off.

5.1 Experimental Setup

We adopt LIBERO as the primary benchmark. LIBERO contains four suites, Spatial, Object, Goal, and Long-Horizon, each comprising 10 language-conditioned robot manipulation tasks and covering spatial relations, object manipulation, goal-conditioned control, and long-horizon tasks. We follow the standard evaluation protocol, executing a fixed number of rollouts per task and using the average **Success Rate (SR)** as the primary task metric.

5.2 Main Results

We use **XXX-WAM** as the base model. All controlled comparisons use the same model weights, visual inputs, action length, number of denoising steps, and numerical precision, changing only the attention computation strategy, so as to isolate the effect of sparsification itself.

In terms of task performance, Sparse-WAM maintains Clean and Randomized success rates close to those of the dense WAM while substantially reducing the amount of attention computation, which indicates that a large amount of visual attention computation in the WAM can be safely pruned.

Compared with Uniform Sparse, Sparse-WAM achieves a higher task success rate under the same computational budget, showing that using a uniform sparsity ratio for different heads and generation stages causes a clear misallocation of the computational budget. Compared with the finer-grained Token-wise Dynamic Sparse, Sparse-WAM achieves a similar or better performance–efficiency trade-off with lower profiling and routing overhead.

Furthermore, Sparse-WAM remains stable in the Randomized environment, which indicates that the learned sparse structure does not simply depend on fixed visual positions but is able to retain the visual information genuinely related to robot manipulation.

In terms of inference efficiency, Sparse-WAM reduces the amount of attention computation by **XXX%**, lowers the attention latency from **XXX ms** to **XXX ms**, and finally reduces the end-to-end inference latency to **XXX ms**, achieving an actual speedup of **XXX**$\times$. This result shows that our method can effectively translate the theoretical benefit of sparse computation into actual WAM inference acceleration.

5.3 Ablation Studies

5.3.1 Dynamic Budget Allocation

Uniform Budget allocates the same amount of computation to all attention heads; Head-wise Budget allocates budgets to different heads according to their statistical characteristics; Head + Stage further takes the differences between generation stages into account.

The results in Table X show that a uniform budget alone causes both redundant computation and pruning of critical dependencies, whereas jointly modeling **head-level and stage-level differences** yields a better performance–efficiency trade-off. Adding the lightweight online adjustment further allows adaptation to the current input variation without paying the high cost of full token-level profiling.

5.3.2 Action-guided Visual Sparsity

We then analyze the role of action information in the sparsification of visual attention.

Using visual-side information alone preserves the basic visual dependencies but cannot sufficiently distinguish visual computation relevant to the current manipulation from that which is irrelevant; using Action-to-Video attention directly as the visual-side sparse mask likewise degrades performance, which indicates that action-side information cannot fully replace the internal Video-to-Video dependencies.

In contrast, taking action information as a prior on the importance of visual computation and further using the internal visual dependencies to supply the necessary context achieves a higher task success rate under the same computational budget. This result validates the core observation of this paper: **action information can guide visual sparsification, but an effective sparse attention for WAMs still needs to preserve the dependency structure of the visual side itself.**

5.3.3 Sparse Execution Efficiency

Finally, we study the effect of different sparse execution schemes on the actual inference speed.

Although fine-grained token-level sparsity can achieve a higher theoretical sparsity ratio, its dynamic indices, routing, and non-contiguous memory access bring additional system overhead, so that the FLOPs reduction cannot be fully translated into actual acceleration. By organizing the dynamic sparse connections into more regular blocks and combining token reordering with low-overhead routing, Sparse-WAM substantially reduces the sparse execution overhead and thereby obtains a higher end-to-end speedup.

6 Conclusion

We presented Sparse-WAM, a decision-aware dynamic sparse attention framework for World Action Models. Starting from the heterogeneity across heads and stages, the complementarity between action guidance and visual dependency, and the tension between sparse structure and actual execution efficiency, we analyzed why sparse attention from video generation is difficult to transfer directly to WAMs, and accordingly designed three modules: action-aware head-wise adaptive budget allocation, action-guided visual sparse routing, and cross-step reuse with efficient block-sparse execution. Together they answer how much to compute, what to compute, and how to compute efficiently in sparse computation for WAMs.

The main limitation of the current work is that task-level quality evaluation still follows the benchmark Success Rate protocol that is being put in place, and the transfer of sparse policies across models and tasks, as well as finer-grained modeling of the sparse structure, remain to be further validated. Future work will focus on rebuilding sparsity labels driven by action impact, sparsity-aware training, and evaluation on a broader range of embodied tasks.

References