STREAMACCEL: AUTOMATED INFERENCE ACCELERATION FOR STREAMING CAUSAL VIDEO MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Interactive video generation requires fast, sustained causal inference to provide timely feedback to user inputs. Yet accelerating a new model still requires substantial manual tuning: practitioners must not only select suitable acceleration techniques but also determine their order and combination. Because earlier optimizations can alter the execution conditions of later techniques, choices that perform best in isolation may not be optimal when combined. Moreover, differences in model structure, streaming state, and deployment make existing configurations difficult to transfer directly, requiring these decisions to be revisited for new models. To address these challenges, we introduce **StreamAccel**, an automated inference optimization system for streaming causal video models. Its staged workflow defines the optimization tasks and prerequisites for each stage, guiding an agent to select, compose, and validate acceleration techniques. Steady-state measurements assess the benefits of these combinations for diffusion-transformer (DiT) generation and VAE decoding. Complementing this workflow, a reusable technique knowledge library captures applicability conditions, composition experience, and failure modes from prior tasks, guiding optimization on new models and enabling engineering experience to be reused across models. Experiments demonstrate that StreamAccel consistently reduces steady-state latency for both generation and decoding across different causal video models.

1 INTRODUCTION

Interactive video generation enables users to explore generated environments and influence their evolution through control inputs (Parker-Holder & Fruchter, 2025; Robbyant Team et al., 2026; Huang et al., 2026; DreamX Team et al., 2026; Mao et al., 2025; Sun et al., 2025; Chen et al., 2026b; Zhang et al., 2026b). To respond to inputs as they arrive, causal video models generate successive chunks conditioned on past context and newly arriving controls (Yin et al., 2025; Huang et al., 2025b). However, the computational cost of generating and decoding each chunk can delay feedback to the user. Efficient inference is therefore essential for timely responses and smooth, sustained interaction.

Turning a new causal model into an efficient implementation still demands substantial performance engineering. A practitioner must understand its streaming path, identify avoidable work, choose applicable optimizations, and determine how to combine them. These choices depend on the model's structure, state representation, and deployment configuration (Li et al., 2026). They also depend on what has already been optimized: an earlier transformation can expose a new bottleneck or change the execution conditions under which a later technique is evaluated. Consequently, neither copying a previous model's fastest configuration nor testing a list of techniques independently provides a reliable optimization procedure. The engineering burden lies in deciding *what to change next, in what order, and when to reconsider an earlier choice*. Without a reusable procedure, this reasoning is repeatedly reconstructed through manual inspection, profiling, implementation, and experimentation for each new model.

Recent work seeks to automate this engineering process through two related routes. The first is *technique-wise tuning followed by global composition*. Sol assigns specialized agents to tune

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.

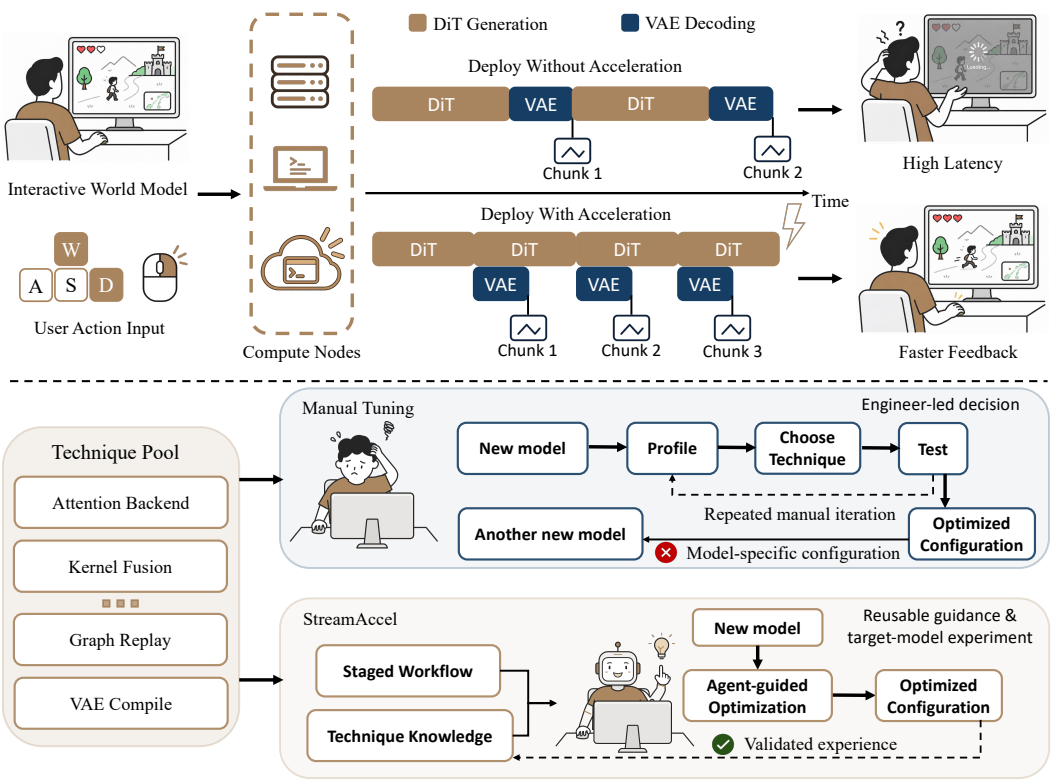

Figure 1: **From manual tuning to reusable streaming acceleration.** Each video chunk requires DiT generation and VAE decoding before its frames reach the user. Whereas manual tuning repeats technique selection for each new model, StreamAccel uses a staged workflow and reusable knowledge to guide model-specific optimization and deliver faster feedback.

caching, sparsity, pruning, quantization, and fusion, then uses an integrator to resolve their interactions under human visual-quality feedback (Li et al., 2026). This moves the optimization task from manually implementing individual techniques to coordinating an acceleration stack. The second is *bottleneck-guided search across the inference stack*. Arbor maintains a tree of tested changes, uses profiling to prioritize subsequent actions, and combines specialist implementation with independent checks of accuracy and measurement validity (Prakriya et al., 2026). Hyperloom brings trace analysis, Arbor-based exploration, and kernel optimization into a staged workflow for inference workloads (AMD, 2026). Both routes reduce manual coordination by making implementation, measurement, and revision part of the optimization loop.

However, Sol's reported experiments focus on video clip generation (Li et al., 2026), while Arbor and Hyperloom evaluate broader inference optimization workflows (Prakriya et al., 2026; AMD, 2026). Systematic optimization of causal video inference remains less explored. Although the computational components remain the DiT and VAE, their streaming execution changes the performance bottlenecks and the benefits of different acceleration techniques (Feng et al., 2026). Effective optimization therefore requires reconsidering which techniques to prioritize and how to combine them for sustained generation and decoding. What remains missing is a reusable optimization workflow that makes these decisions systematically for new causal video models.

Our central question is: *How can we systematically identify effective combinations of optimizations for new causal video models, while preserving generation quality and interactive behavior and reducing reliance on model-by-model manual tuning?*

To this end, we introduce **StreamAccel**, an automated inference optimization system for streaming causal video models. StreamAccel organizes acceleration techniques into a dependency-aware, staged workflow that accounts for the execution and state characteristics of causal inference. Within this framework, steady-state profiling identifies bottlenecks in the current implementation and guides

candidate selection, while integrated evaluation determines whether candidate transformations improve performance in combination with existing optimizations. To make this process reusable across models, StreamAccel maintains a technique knowledge library of applicability conditions, composition lessons, and failure modes, transferring engineering guidance rather than previously optimized configurations. By combining a structured workflow, workload-specific performance evidence, and reusable knowledge, StreamAccel systematically explores acceleration strategies for sustained generation and decoding, with visual-quality and causal-behavior checks to assess preservation of the model's generative and interactive capabilities.

We evaluate StreamAccel on Solaris and SolarWM-Wan5B. StreamAccel reduces steady-state DiT latency by 38.33% and 32.21%, respectively, with corresponding VAE latency reductions of 24.84% and 3.79%. These results demonstrate its effectiveness in accelerating causal video inference.

Our contributions are:

- **Automated acceleration framework.** We introduce StreamAccel, a system that organizes technique selection and composition into a dependency-aware, staged workflow for streaming causal video generation and decoding.
- **Reusable optimization knowledge.** We develop a technique knowledge library that transfers applicability conditions and composition experience across models, guiding optimization without prescribing model-specific outcomes.
- **Cross-model acceleration.** Across Solaris and SolarWM-Wan5B, we reduce DiT latency by 32.21%–38.33% and VAE latency by up to 24.84%, while maintaining visual quality and measured control response in paired long-rollout evaluations.

2 Related Work

2.1 Video generation and interactive world models

Early video diffusion models generate coherent clips by jointly denoising frames (Ho et al., 2022b; Singer et al., 2023; Ho et al., 2022a). Stable Video Diffusion, CogVideoX, HunyuanVideo, and Wan follow this approach in latent space with bidirectional temporal context (Blattmann et al., 2023; Yang et al., 2025; Kong et al., 2024; Team Wan et al., 2025). This couples generation across the clip and can delay responses to incoming inputs. Causal models instead generate chunks from history and current inputs: CausVid distills a bidirectional teacher into a causal student (Yin et al., 2025). Self Forcing and related work further study how to extend autoregressive video generation and improve quality and consistency over long sequences (Huang et al., 2025b; Cui et al., 2025; Yang et al., 2026; Liu et al., 2026b; Yi et al., 2026; Chen et al., 2026a; Huang et al., 2025a).

Interactive world models combine incremental generation with user control, enabling exploration of generated environments (Parker-Holder & Fruchter, 2025; He et al., 2025; Robbyant Team et al., 2026; DreamX Team et al., 2026; Mao et al., 2025; Zhu et al., 2026; Huang et al., 2026). EchoWM extends this interaction to synchronized audio and video (Zhang et al., 2026b). For sustained interaction, WorldPlay, ReWorld, and Matrix-Game 3.0 introduce memory mechanisms to maintain scene consistency (Sun et al., 2025; Chen et al., 2026b; Skywork AI Matrix-Game Team, 2026). These models expand the capabilities of interactive generation, while efficient deployment also requires accelerating generation and decoding. StreamAccel complements these advances by automating implementation-level optimization of existing causal video models.

2.2 Video inference acceleration and automated optimization

Existing work accelerates video generation at three levels: computational workload, operator efficiency, and system execution. At the workload level, feature reuse and prediction reduce repeated computation across denoising steps (Liu et al., 2025a;b), while sparse attention and token-wise computation allocation reduce work within denoising steps (Xi et al., 2025; Liu et al., 2026a). Generator and decoder architecture design further improves efficiency at the model level (Chen et al., 2025; Kang & Kwak, 2026). For the remaining computation, efficient attention kernels improve memory access and work organization, while quantization lowers costs through low-precision arithmetic (Dao et al., 2022; Dao, 2024; Zhang et al., 2025; Zhao et al., 2025). System-level methods further

organize computation through multi-device parallelism and streaming scheduling (Fang et al., 2024; Feng et al., 2026). These techniques operate at different levels, and their applicability and combined gains depend on the model and deployment conditions (Li et al., 2026). Beyond developing individual techniques, inference optimization therefore also requires selecting, implementing, and composing them for a specific target.

Automated optimization systems aim to reduce this manual tuning effort by using execution feedback to guide code changes. This research spans kernel generation and evaluation, as studied by KernelBench (Ouyang et al., 2025), experience-guided kernel optimization in AccelOpt (Zhang et al., 2026a), and cross-layer search and reusable optimization recipes in Arbor and Hyperloom (Prakriya et al., 2026; AMD, 2026). For video generation, Sol directly addresses technique composition: specialized agents tune individual technique families, and an integrator combines them with human feedback on generation quality (Li et al., 2026). Sol's experiments primarily evaluate video clip generation, whereas StreamAccel targets sustained generation and streaming decoding in causal video models. For this workload, StreamAccel uses a staged workflow to organize the selection, implementation, and composition of these acceleration techniques. Its technique library records applicability conditions and composition experience to guide optimization of new models.

3 Background and Design Motivation

3.1 Streaming Execution and Optimization Opportunities

Latent video generation denoises in latent space and decodes the resulting latents into visible frames (Rombach et al., 2022; Peebles & Xie, 2023). Causal models repeat this process over successive chunks: each chunk uses retained history and current conditioning to generate new content, then updates the state needed by future chunks (Yin et al., 2025; Huang et al., 2025b). Figure 2 illustrates this recurring execution. For chunk t, we summarize the work as

$$z_t = D_\theta(\epsilon_t, u_t; s_t), \qquad s_{t+1} = R_\theta(s_t, z_t, u_t), \qquad (x_t, v_{t+1}) = V_\theta(z_t, v_t). \tag{1}$$

Here D_θ generates latents z_t from noise ϵ_t, current conditioning u_t, and generation history s_t. The generated latents then enter two computations: R_θ updates the history used by later chunks, and V_θ decodes new frames x_t while updating decoder state v_t.

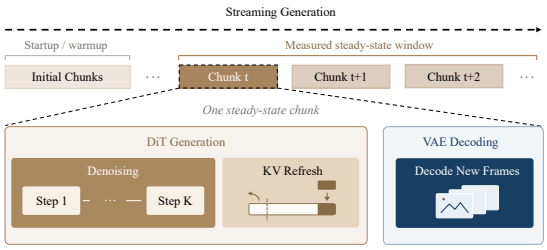

Figure 2: **Streaming execution.** DiT denoising, KV refresh, and VAE decoding recur per chunk.

This execution pattern creates both an opportunity and a constraint. DiT generation and VAE decoding recur for every new chunk, so overhead on the repeated path accumulates throughout a session. Yet the model does not rerun a fixed-input computation: history and decoder state evolve, controls can change, and the state left by one chunk shapes later outputs. Acceleration must therefore lower recurring cost without disrupting the evolution of history or the model's response to new controls.

3.2 Design Rationale for Staging and Knowledge Reuse

The order of some optimization decisions is consequential. Certain techniques require a prepared execution or state path, while earlier changes can alter the conditions under which later ones are measured. For example, compilation can fuse neighboring operations and change the relative cost of attention backends (Ansel et al., 2024). Graph capture likewise depends on first identifying the boundaries of recurring computation and history updates. A staged workflow therefore arranges the prerequisites for experiments without prescribing which techniques will form the final configuration.

Ordering alone does not make individual gains additive. Two techniques may remove the same overhead or change one another's execution, so the best choices in isolation need not form the fastest combination. Small experiments can check feasibility and expose costs, but adoption depends on measuring a candidate within the current configuration. If a later change alters an earlier choice's assumptions, that choice requires targeted revalidation rather than an unrestricted restart of the search.

The fastest configuration also depends on the model, hardware, and workload and cannot be assumed to transfer directly between models (Li et al., 2026). What can transfer is engineering knowledge: where a technique applies, how it interacts with others, why it may fail, and how to validate it. Such experience provides a targeted starting point for a new model, not a substitute for target-model measurements. StreamAccel therefore uses a staged workflow and technique knowledge library to guide optimization. Tests on the current configuration determine which changes to retain.

4 StreamAccel

4.1 System overview

StreamAccel targets the sustained inference of existing causal video models. For each new chunk, the DiT denoises latents and updates the history needed for later generation, and the streaming VAE decodes the new latents into visible frames. Given a model, hardware setup, and streaming workload, StreamAccel aims to reduce the steady-state cost of generation and decoding while preserving video quality and control response. Because implementations and bottlenecks differ across models, it does not prescribe a universally optimal acceleration configuration.

Instead, StreamAccel first establishes an executable reference for the target model's streaming inference and then follows the stages described in Section 4.2. Each stage specifies what to investigate and which checks are required, but does not prescribe specific code changes. The agent derives candidate changes from the target model's code and measurements. The technique library (Section 4.3) supplies relevant applicability conditions and lessons from past failures, rather than a configuration copied from another model. As detailed in Section 4.4, performance-sensitive changes first receive a focused trial. Their steady-state benefit and generation behavior are then evaluated in combination with the optimizations already retained. A change that passes these checks becomes the starting point for subsequent experiments. If it does not pass, subsequent trials continue from the unchanged implementation. Once a stage is complete, the workflow advances to the next. The resulting implementation is supported by performance and behavior evidence. Figure 3 summarizes the process.

4.2 Dependency-aware workflow

StreamAccel's stages are not a universal checklist of acceleration tricks. Instead, they establish the execution conditions under which a technique can be evaluated. An earlier change may alter the computation seen by later techniques: compilation, for example, changes the operations surrounding attention, so a backend ranking obtained before compilation need not hold afterward (Ansel et al., 2024). The workflow therefore first establishes a stable streaming execution path, then evaluates techniques that depend on that path.

Starting from the model's native streaming inference, the workflow establishes a comparable reference and prepares the DiT computation repeated for each chunk. This is more than code cleanup: it distinguishes work that can be reused from state that must change as generation proceeds. For example, the cross-attention projections of unchanged conditioning and RoPE frequency tables for a known position range can be reused, whereas current positions, incoming controls, and history-cache writes must still update. The workflow also removes avoidable compiler graph breaks. Together, these changes reduce repeated work and let compilation cover a larger region, creating more opportunities for operator fusion.

On this prepared path, the next stage establishes stable DiT compilation. The workflow then evaluates kernels for self- and cross-attention at their actual sites, shapes, and execution conditions, selecting a suitable kernel for each rather than imposing one backend throughout the model. Only after compilation and attention selection does it profile the steady-state implementation. The resulting bottlenecks are those that remain after these optimizations, rather than costs already removed from the original code.

Subsequent experiments target the costs exposed by that profile. When redundant data preparation or operator work remains, StreamAccel tests implementations that remove the excess work and expose further fusion. When repeated kernel launches remain costly, it explores CUDA Graph replay to reduce submission overhead (Nguyen et al., 2021). Both decisions are made on the compiled implementation, where some of the original work may already have been removed or fused. If

a later change alters the conditions behind an earlier choice, StreamAccel rechecks the affected choice rather than restarting the entire search.

Once DiT generation is settled, StreamAccel optimizes native streaming VAE decoding through applicable compilation and operator changes. These changes must preserve temporal-cache updates across chunks, because accelerating an isolated, stateless decode would not represent sustained decoding. The resulting generation and decoding paths are then checked in sustained streaming inference. Appendix A details the stage tasks and evidence requirements.

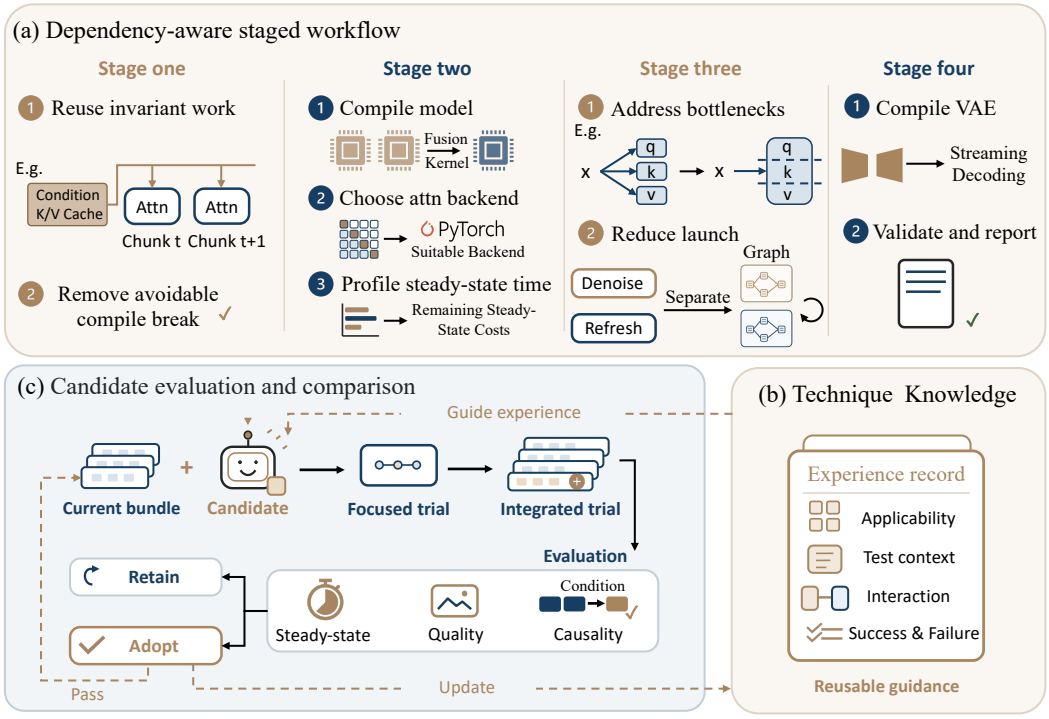

Figure 3: **StreamAccel overview.** (a)Dependency-aware stages organize the optimization of DiT generation and streaming VAE decoding. (b) The technique library provides conditional guidance for designing candidates. (c) Focused trials examine candidate changes, while integrated evaluation measures steady-state latency and checks generation behavior before adoption.

4.3 Technique Knowledge Across Models

The same optimization can behave differently across models, or even across sites within one model. StreamAccel therefore retains not only which techniques were used, but also why they were chosen. Its technique library records how a technique is implemented and when it applies, together with the execution conditions under which it was tested, its interactions with other optimizations, observed failures, and the experiments that exposed them. At each stage, the agent checks relevant records against the target code, hardware, and current optimization stack. This experience narrows the search and helps avoid known pitfalls, but does not predetermine the fastest configuration for a new model.

Attention-backend selection in Solaris illustrates this distinction. Cross-attention in its action module operates on relatively small matrices, so its best backend need not be the one used for self-attention during DiT denoising. On the compiled NVIDIA H20 execution path, we compared three scaled dot-product attention backends: FlashAttention (a fused attention implementation), MATH (PyTorch's mathematical implementation), and EFFICIENT (a memory-efficient fused implementation). MATH outperformed the default EFFICIENT path at this action site and was retained after full-model and behavior checks (Appendix A.2). The transferable lesson is not that every action-attention layer should use MATH. Rather, attention site, computation shape, hardware, and compilation can all change the backend ranking. For a new model, this record prompts the agent to compare

eligible backends at the relevant sites, including any required data-layout conversions, instead of assigning one kernel to the entire model.

Graph capture offers a second kind of experience. One strategy is to attempt the largest semantically safe capture region first, then progressively narrow it—from the whole model or an execution phase to groups of DiT blocks, or even a single block—if a larger capture is unsafe or impractical. Another concerns state updates: when denoising repeatedly reads history but KV refresh commits new context only once per chunk, the two phases require separate capture decisions. Including that commit in a graph replayed at every denoising step would update history repeatedly. The corresponding checks verify that each replay uses current positions and controls and leaves valid history for the next chunk. Together, these lessons help the agent choose capture boundaries and diagnose failures.

4.4 Candidate evaluation and composition

StreamAccel stores accepted code changes, runtime configuration, and their supporting evidence as a *bundle*. At the end of each stage, it freezes the resulting bundle as the starting point for the next. Within a stage, the agent draws candidates from the target code, measurements, and technique library. A focused small experiment first checks whether a change is feasible and whether its intended mechanism works. Such experiments can expose implementation problems quickly, but cannot establish the benefit of a change once it interacts with existing optimizations. Any candidate considered for adoption must therefore be integrated into the current bundle and evaluated on the full streaming inference path.

Integrated evaluation first asks whether the candidate improves steady-state performance. Under the same workload, StreamAccel compares the current bundle with a version containing the candidate, measuring DiT generation and streaming VAE decoding time per chunk during sustained execution. Timing includes work repeated for each chunk, such as required cache updates and decoder-state maintenance. One-time compilation, graph capture, and warmup are excluded. This full-path comparison measures the candidate's benefit in the existing optimization stack, rather than the speed of an isolated operator.

Integrated evaluation also checks whether generation behavior is preserved. Checks follow the state affected by the change: a history-cache modification requires continuation and request-reset tests, while a control-path change must leave the generated prefix unchanged and still affect subsequent output. The retained full stack is also examined over long videos for perceptible visual degradation.

A candidate enters a new bundle only if it meets the performance criterion and passes the relevant behavior checks. Each decision retains its test conditions and supporting evidence. If a later change invalidates the conditions behind an earlier choice, StreamAccel rechecks only the affected decisions. Reviewed lessons can then enter the technique library to guide subsequent models.

5 Experiments

In this section, we evaluate StreamAccel through two questions. **Q1:** How much does StreamAccel reduce the steady-state cost of DiT generation and VAE decoding across causal video models? **Q2:** How does acceleration affect the visual quality and control behavior of long rollouts? We first define the models, matched workloads, and metrics (Section 5.1), then answer Q1 with latency measurements (Section 5.2) and Q2 with paired quality and control evaluations (Section 5.3).

5.1 Experimental Setup

Models. Solaris is an action-conditioned Minecraft world model supporting causal rollout and multiplayer views (Savva et al., 2026). We evaluate its causal inference path at 640×352 resolution. SolarWM-Wan5B is a camera-conditioned, causal video world model built on Wan2.2 (Huang et al., 2026). We evaluate its 5B configuration. Both models use their original weights, BF16 precision, four denoising steps, and one NVIDIA H20, without tensor or sequence parallelism.

Optimization agent. The StreamAccel harness used GPT-5.6 Sol with `high` reasoning effort. Appendix A.1 describes this agent configuration and distinguishes optimization runs from evaluation of the retained implementations.

Performance metrics. After warmup, we measure mean GPU time per chunk during steady-state generation. We report DiT generation (denoising and KV-cache refresh) and streaming VAE decoding separately. Each model is compared with its own Reference configuration under a matched workload. Appendix B specifies the timing windows and aggregation.

Quality metrics. For visual quality, we compare 60 paired long rollouts per model using five VBench dimensions: subject and background consistency, aesthetic and imaging quality, and motion smoothness (Huang et al., 2024). We score 128-frame windows from the early, middle, and late portions of each rollout to capture changes over time. Visual metrics do not measure response to user input, so we assess control with a measure suited to each model. Solaris takes discrete steering actions, which we correlate with the resulting scene motion. For the camera-controlled SolarWM-Wan5B, we use WBench NavScore to evaluate how the estimated camera motion follows the requested navigation (Ying et al., 2026). Appendix C explains how we compute both control measures and match Reference–StreamAccel outputs.

Table 1: **Visual quality and control response.** Reference and StreamAccel are compared using VBench visual metrics and model-specific control measures. Higher values indicate better performance, and Δ denotes StreamAccel minus Reference before rounding.

Configuration	Subject ↑	Background ↑	Aesthetic ↑	Imaging ↑	Motion ↑	Control
Solaris *(control: steering–scene correlation ↑)*						
Reference	0.8644	0.9506	0.5381	66.041	0.9575	0.1670
StreamAccel	0.8636	0.9503	0.5359	66.040	0.9562	0.1697
Δ	−0.0009	−0.0003	−0.0022	−0.001	−0.0013	+0.0027
SolarWM-Wan5B *(control: WBench NavScore ↑)*						
Reference	0.8373	0.9169	0.4880	60.256	0.9703	0.8220
StreamAccel	0.8458	0.9205	0.4983	60.988	0.9697	0.8222
Δ	+0.0085	+0.0035	+0.0102	+0.732	−0.0006	+0.0002

5.2 Steady-State Acceleration

Figure 4 compares steady-state DiT generation and streaming VAE decoding time per chunk before and after acceleration. On Solaris, StreamAccel reduces DiT time by 38.33% (289.0 to 178.2 ms) and VAE time by 24.84% (130.9 to 98.4 ms). On SolarWM-Wan5B, it reduces DiT time by 32.21% (1349.1 to 914.5 ms) and VAE time by 3.79% (1280.7 to 1232.1 ms). StreamAccel thus delivers substantial DiT gains on both models, while the benefit to VAE decoding is more model-dependent.

To understand the DiT gains, we separate denoising from KV-cache refresh. Denoising time falls by 38.17% on Solaris and 38.84% on SolarWM-Wan5B. The corresponding refresh reductions are 38.98% and 7.20%. Thus, the two models obtain similar denoising gains, but the smaller refresh gain on SolarWM-Wan5B limits its overall DiT improvement.

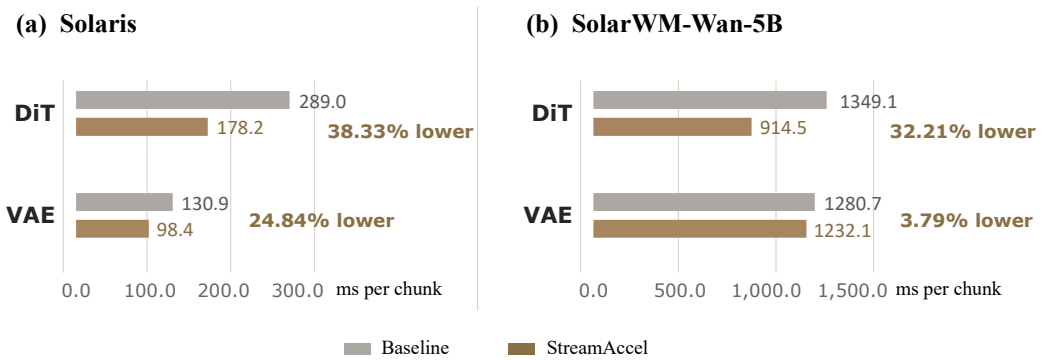

Figure 4: **Steady-state DiT and VAE time per chunk.** Reference and StreamAccel run matched work within each model on one H20.

5.3 Visual Quality and Control Response

Table 1 compares Reference and StreamAccel on visual quality and model-specific control measures. Solaris stays close to its reference across all five VBench dimensions. SolarWM-Wan5B improves on four dimensions, with only a small decrease in motion smoothness. The control measures are similarly stable: Solaris steering–scene correlation changes from 0.1670 to 0.1697, and SolarWM-Wan5B NavScore from 0.8220 to 0.8222 (Ying et al., 2026). Thus, the latency gains in Figure 4 are accompanied by closely matched visual scores and stable control measures. Additional control diagnostics and the evaluation scope are given in Appendix C.

Figure 5 shows three paired SolarWM-Wan5B rollouts at five points spanning each video. Across the shown cases, StreamAccel produces videos with the same overall visual quality and coherent progression as the unaccelerated references. Together with the aggregate scores in Table 1, these comparisons support preserved visual quality and measured control response in the tested rollouts.

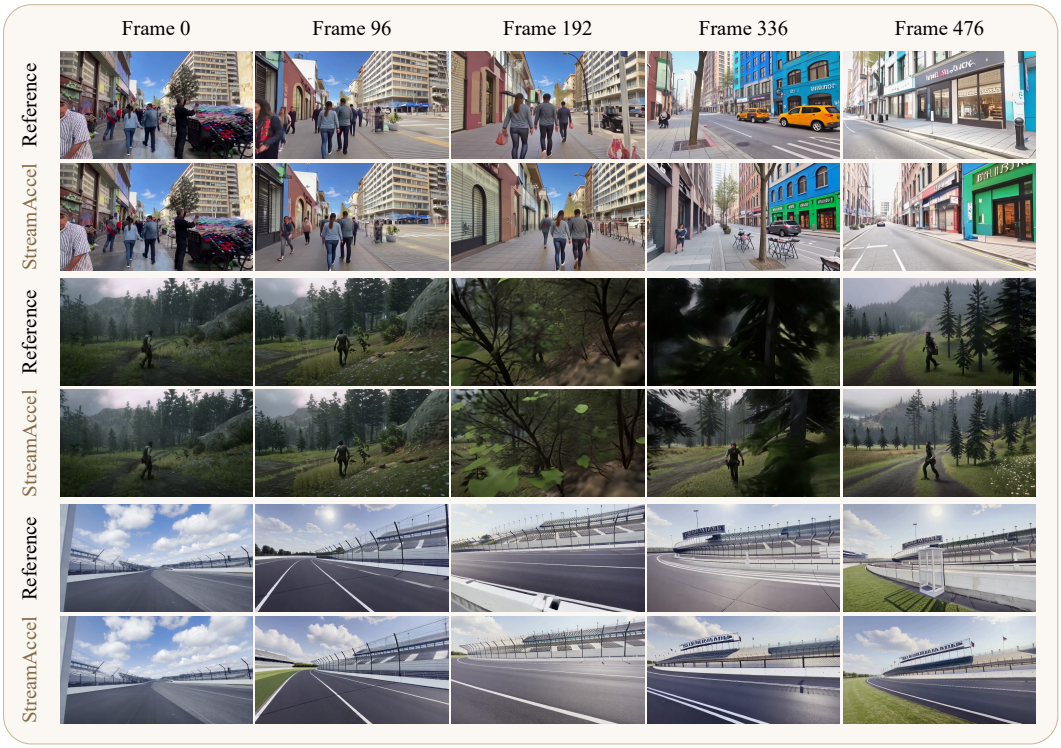

Figure 5: **SolarWM-Wan5B visual comparison.** Reference and StreamAccel rollouts are shown at five frame indices. Additional visual comparisons are shown in Appendix C.3.

6 Conclusion

StreamAccel provides a reusable way to optimize sustained inference in causal video models. Its staged workflow prepares the streaming execution path before testing and composing acceleration techniques, while its technique library guides new experiments without imposing a fixed configuration. On Solaris and SolarWM-Wan5B, it reduces steady-state DiT time per chunk by 38.33% and 32.21%, respectively, and streaming VAE time by 24.84% and 3.79%. Paired long-rollout evaluations show closely matched visual quality, with stable model-specific control measures. Together, these results show that a reusable optimization workflow can deliver substantial DiT gains across distinct causal models while keeping the tested visual and interactive behavior close to the references. Future work will quantify optimization effort and evaluate a wider range of causal architectures.

AI USE STATEMENT

Generative AI tools assisted with implementing parts of the optimization workflow, planning and analyzing experiments, searching the literature, an HTML page, and polishing the manuscript. StreamAccel itself uses an AI agent to propose and implement optimization candidates, which are retained only after model-specific experiments support their adoption. The authors reviewed the reported evidence and AI-assisted content and take responsibility for the final paper.

ETHICS STATEMENT

StreamAccel accelerates existing video generators rather than training a new model. Lower latency can improve interactive applications, but it may also make misleading synthetic video easier to produce. Our quality and control tests do not assess these broader risks. Deployments should preserve the safeguards and usage restrictions of the underlying models.

REPRODUCIBILITY STATEMENT

Section 5.1 specifies the evaluated models, hardware, matched workloads, and metrics, while Section 5.2 reports the latency results. Appendix A documents the stage protocol, agent setting, and a representative attention-backend decision. Appendices B and C describe the timing boundaries, aggregation, paired rollouts, and visual and control evaluations behind the reported results.

REFERENCES

AMD. ROCm Hyperloom: Agentic inference optimization. https://github.com/AMD-AGI/Hyperloom, 2026. Official software and optimization-loop documentation, accessed September 12, 2026.

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 2024. doi: 10.1145/3620665.3640366. URL https://docs.pytorch.org/assets/pytorch2-2.pdf.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. Stable Video Diffusion: Scaling latent video diffusion models to large datasets. https://arxiv.org/abs/2311.15127, 2023. arXiv:2311.15127.

Jiayu Chen, Junbei Tang, Wenbiao Zhao, Maoliang Li, Jiayi Luo, Zihao Zheng, Jiawei Yang, Guojie Luo, and Xiang Chen. Pyramid Forcing: Head-aware pyramid KV cache policy for high-quality long video generation. https://arxiv.org/abs/2605.13111, 2026a. arXiv:2605.13111.

Junsong Chen, Yuyang Zhao, Jincheng Yu, Ruihang Chu, Junyu Chen, Shuai Yang, Xianbang Wang, Yicheng Pan, Daquan Zhou, Huan Ling, Haozhe Liu, Hongwei Yi, Hao Zhang, Muyang Li, Yukang Chen, Han Cai, Sanja Fidler, Ping Luo, Song Han, and Enze Xie. SANA-Video: Efficient video generation with block linear diffusion transformer. https://arxiv.org/abs/2509.24695, 2025. arXiv:2509.24695.

Zhifei Chen, Luozhou Wang, Guibao Shen, Dongyu Yan, Shuai Yang, Tianshuo Xu, Yihua Du, Wei Wang, Tianyi Gui, Lianghua Huang, and Yingcong Chen. ReWorld: An interactive world model with long-horizon memory. https://arxiv.org/abs/2608.23565, 2026b. arXiv:2608.23565.

Justin Cui, Jie Wu, Ming Li, Tao Yang, Xiaojie Li, Rui Wang, Andrew Bai, Yuanhao Ban, and Cho-Jui Hsieh. Self-Forcing++: Towards minute-scale high-quality video generation. https://arxiv.org/abs/2510.02283, 2025. arXiv:2510.02283.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2307.08691.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2205.14135.

DreamX Team, Yancheng Bai, Rui Chen, Xiangxiang Chu, Rujing Dang, Hao Dou, Bingjie Gao, Qiwen Gu, Siyu Hong, Jiachen Lei, Geng Li, Jifan Li, Ruimin Lin, Qingfeng Shi, Bingze Song, Lei Sun, Jing Tang, Ruitian Tian, Jun Wang, Jiahong Wu, Pengfei Zhang, Shen Zhang, and Jiashu Zhu. DreamX-World 1.0: A general-purpose interactive world model. https://arxiv.org/abs/2606.16993, 2026. arXiv:2606.16993.

Jiarui Fang, Jinzhe Pan, Xibo Sun, Aoyu Li, and Jiannan Wang. xDiT: An inference engine for diffusion transformers (DiTs) with massive parallelism. https://arxiv.org/abs/2411.01738, 2024. arXiv:2411.01738.

Tianrui Feng, Zhi Li, Shuo Yang, Haocheng Xi, Muyang Li, Xiuyu Li, Lvmin Zhang, Keting Yang, Kelly Peng, Song Han, Maneesh Agrawala, Kurt Keutzer, Akio Kodaira, and Chenfeng Xu. StreamDiffusionV2: A streaming system for dynamic and interactive video generation. *Proceedings of Machine Learning and Systems*, 8:1656–1669, 2026. URL https://streamdiffusionv2.github.io/.

Xianglong He, Chunli Peng, Zexiang Liu, Boyang Wang, Yifan Zhang, Qi Cui, Fei Kang, Biao Jiang, Mengyin An, Yangyang Ren, Baixin Xu, Hao-Xiang Guo, Kaixiong Gong, Size Wu, Wei Li, Xuchen Song, Yang Liu, Yangguang Li, and Yahui Zhou. Matrix-Game 2.0: An open-source real-time and streaming interactive world model. https://arxiv.org/abs/2508.13009v4, 2025. arXiv:2508.13009v4; first released in 2025, revised in 2026.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen Video: High definition video generation with diffusion models. https://arxiv.org/abs/2210.02303, 2022a. arXiv:2210.02303.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. In *Advances in Neural Information Processing Systems*, volume 35, 2022b. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/39235c56aef13fb05a6adc95eb9d8d66-Abstract-Conference.html.

Junchao Huang, Xinting Hu, Boyao Han, Shaoshuai Shi, Zhuotao Tian, Tianyu He, and Li Jiang. Memory Forcing: Spatio-temporal memory for consistent scene generation on Minecraft. https://arxiv.org/abs/2510.03198, 2025a. arXiv:2510.03198.

Junchao Huang, Guian Fang, Shengju Qian, Xianghao Kong, Zhuoran Zhao, Wei Huang, Yihua Du, Zixin Zhang, Justin Cui, Yuchao Gu, Yukang Chen, Xinting Hu, Tianyu He, Shaoshuai Shi, Zhuotao Tian, Xin Wang, Mike Zheng Shou, and Li Jiang. SolarWM: Open data and scalable training for long-horizon video world models. https://arxiv.org/abs/2609.02886, 2026. arXiv:2609.02886.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. In *Advances in Neural Information Processing Systems*, 2025b. URL https://neurips.cc/virtual/2025/poster/116208.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video

generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21807–21818, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Huang_VBench_Comprehensive_Benchmark_Suite_for_Video_Generative_Models_CVPR_2024_paper.html.

Minguk Kang and Suha Kwak. FlashDecoder: Real-time latent-to-pixel streaming decoder with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. URL https://arxiv.org/abs/2607.14898.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, Kathrina Wu, Qin Lin, Junkun Yuan, Yanxin Long, Aladdin Wang, Andong Wang, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Hongmei Wang, Jacob Song, Jiawang Bai, Jianbing Wu, Jinbao Xue, Joey Wang, Kai Wang, Mengyang Liu, Pengyu Li, Shuai Li, Weiyan Wang, Wenqing Yu, Xinchi Deng, Yang Li, Yi Chen, Yutao Cui, Yuanbo Peng, Zhentao Yu, Zhiyu He, Zhiyong Xu, Zixiang Zhou, Zunnan Xu, Yangyu Tao, Qinglin Lu, Songtao Liu, Dax Zhou, Hongfa Wang, Yong Yang, Di Wang, Yuhong Liu, Jie Jiang, and Caesar Zhong. HunyuanVideo: A systematic framework for large video generative models. https://arxiv.org/abs/2412.03603, 2024. arXiv:2412.03603.

Yitong Li, Junsong Chen, Haopeng Li, Haozhe Liu, Jincheng Yu, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Sol video inference engine: Agent-native full-stack acceleration framework for efficient video generation. https://arxiv.org/abs/2606.23743v2, 2026. arXiv:2606.23743v2.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep Embedding Tells: It's Time to Cache for Video Diffusion Model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025a.

Haosong Liu, Yuge Cheng, Wenxuan Miao, Zihan Liu, Aiyue Chen, Jing Lin, Yiwu Yao, Chen Chen, Jingwen Leng, Minyi Guo, and Yu Feng. Astraea: A token-wise acceleration framework for video diffusion transformers. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=e8P4Oo8S6U.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with TaylorSeers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025b. URL https://arxiv.org/abs/2503.06923v2.

Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling Forcing: Autoregressive long video diffusion in real time. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://arxiv.org/abs/2509.25161.

Xiaofeng Mao, Shaoheng Lin, Zhen Li, Chuanhao Li, Wenshuo Peng, Tong He, Jiangmiao Pang, Mingmin Chi, Yu Qiao, and Kaipeng Zhang. Yume: An interactive world generation model. https://arxiv.org/abs/2507.17744, 2025. arXiv:2507.17744.

Vinh Nguyen, Michael Carilli, Sukru Burc Eryilmaz, Vartika Singh, Michelle Lin, Natalia Gimelshein, Alban Desmaison, and Edward Yang. Accelerating PyTorch with CUDA Graphs. https://pytorch.org/blog/accelerating-pytorch-with-cuda-graphs/, 2021. PyTorch technical article, October 26. Accessed September 12, 2026.

Anne Ouyang, Simon Guo, Simran Arora, Alex L. Zhang, William Hu, Christopher Ré, and Azalia Mirhoseini. KernelBench: Can LLMs write efficient GPU kernels? https://arxiv.org/abs/2502.10517, 2025. arXiv:2502.10517.

Jack Parker-Holder and Shlomi Fruchter. Genie 3: A new frontier for world models. https://deepmind.google/blog/genie-3-a-new-frontier-for-world-models/, 2025. Google DeepMind technical article, August 5.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. URL https://arxiv.org/abs/2212.09748.

Neha Prakriya, Chaojun Hou, Zheng Gong, Huasha Zhao, Xi Zhao, Mou Li, Zhenyu Gu, and Emad Barsoum. Arbor: Tree search as a cognition layer for autonomous agents. https://arxiv.org/abs/2606.12563v1, 2026. arXiv:2606.12563v1.

Robbyant Team, Zelin Gao, Qiuyu Wang, Yanhong Zeng, Jiapeng Zhu, Ka Leong Cheng, Yixuan Li, Hanlin Wang, Yinghao Xu, Shuailei Ma, Yihang Chen, Jie Liu, Yansong Cheng, Yao Yao, Jiayi Zhu, Yihao Meng, Kecheng Zheng, Qingyan Bai, Jingye Chen, Zehong Shen, Yue Yu, Xing Zhu, Yujun Shen, and Hao Ouyang. Advancing open-source world models. https://arxiv.org/abs/2601.20540, 2026. arXiv:2601.20540.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. URL https://arxiv.org/abs/2112.10752.

Georgy Savva, Oscar Michel, Daohan Lu, Suppakit Waiwitlikhit, Timothy Meehan, Dhairya Mishra, Srivats Poddar, Jack Lu, and Saining Xie. Solaris: Building a multiplayer video world model in Minecraft. https://arxiv.org/abs/2602.22208, 2026. arXiv:2602.22208.

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-A-Video: Text-to-video generation without text-video data. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://iclr.cc/virtual/2023/poster/11001.

Skywork AI Matrix-Game Team. Matrix-Game 3.0: Real-time and streaming interactive world model with long-horizon memory. https://github.com/SkyworkAI/Matrix-Game/blob/main/Matrix-Game-3/assets/pdf/report.pdf, 2026. Technical report.

Wenqiang Sun, Haiyu Zhang, Haoyuan Wang, Junta Wu, Zehan Wang, Zhenwei Wang, Yunhong Wang, Jun Zhang, Tengfei Wang, and Chunchao Guo. WorldPlay: Towards long-term geometric consistency for real-time interactive world modeling. https://arxiv.org/abs/2512.14614, 2025. arXiv:2512.14614.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. https://arxiv.org/abs/2503.20314, 2025. arXiv:2503.20314.

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, Jianfei Chen, Ion Stoica, Kurt Keutzer, and Song Han. Sparse VideoGen: Accelerating video diffusion transformers with spatial-temporal sparsity. In *Proceedings of the International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2502.01776v2.

Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, Song Han, and Yukang Chen. LongLive: Real-time interactive long video generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.22622.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. CogVideoX: Text-to-video diffusion models with an expert transformer. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2408.06072v3.

Jung Yi, Wooseok Jang, Paul Hyunbin Cho, Jisu Nam, Heeji Yoon, and Seungryong Kim. Deep Forcing: Training-free long video generation with deep sink and participative compression. In *Proceedings of the International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/62407.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T. Freeman, Frédo Durand, Eli Shechtman, and Xun Huang. From Slow Bidirectional to Fast Autoregressive Video Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Kaining Ying, Hengrui Hu, Siyu Ren, Jiamu Li, Fengjiao Chen, Ziwen Wang, Xuezhi Cao, Xunliang Cai, and Henghui Ding. WBench: A comprehensive multi-turn benchmark for interactive video world model evaluation. https://arxiv.org/abs/2605.25874, 2026. arXiv:2605.25874.

Genghan Zhang, Shaowei Zhu, Anjiang Wei, Zhenyu Song, Allen Nie, Zhen Jia, Nandita Vijaykumar, Yida Wang, and Kunle Olukotun. AccelOpt: A self-improving LLM agentic system for AI accelerator kernel optimization. In *Proceedings of Machine Learning and Systems*, volume 8, pp. 541–568. MLSys, 2026a. URL https://proceedings.mlsys.org/paper_files/paper/2026/file/0f8426558905746fc38da5e335700aec-Paper-Conference.pdf.

Jintao Zhang, Jia Wei, Pengle Zhang, Jun Zhu, and Jianfei Chen. SageAttention: Accurate 8-bit attention for plug-and-play inference acceleration. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/b286c344d38e10d2466c0514b78e2f36-Paper-Conference.pdf.

Songchun Zhang, Yaowei Li, Junhao Zhuang, Weiyang Jin, Haoyu Wang, Xin Lu, Yilang Sun, Shiyi Zhang, Haoran Li, Xiaoxiao Ma, Yuming Li, Yijun Liu, Yaofeng Su, Yanwen Ma, Haoyu Wu, Zihan Su, Yue Ma, Lvmin Zhang, Haoyang Huang, Zeyue Xue, Anyi Rao, and Nan Duan. EchoWM: Open and enterable omnimodal world models. https://arxiv.org/abs/2608.23189, 2026b. arXiv:2608.23189.

Tianchen Zhao, Tongcheng Fang, Haofeng Huang, Enshu Liu, Rui Wan, Widyadewi Soedarmadji, Shiyao Li, Zinan Lin, Guohao Dai, Shengen Yan, Huazhong Yang, Xuefei Ning, and Yu Wang. ViDiT-Q: Efficient and accurate quantization of diffusion transformers for image and video generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2406.02540v3.

Haoyi Zhu, Haozhe Liu, Yuyang Zhao, Tian Ye, Junsong Chen, Jincheng Yu, Tong He, Song Han, and Enze Xie. SANA-WM: Efficient minute-scale world modeling with hybrid linear diffusion transformer. https://arxiv.org/abs/2605.15178, 2026. arXiv:2605.15178.

A STAGE PROTOCOL AND REUSABLE EXPERIENCE

The stages in Section 4.2 specify what the agent must establish before a class of changes can be judged. They do not prescribe a backend or kernel for every model. Table 2 gives the stage tasks and the evidence needed to advance. Every stage integrates promising changes into its current bundle and checks steady-state performance and affected behavior before accepting them. Validation is therefore part of every stage rather than a separate final step. DiT generation is addressed before native streaming VAE decoding. A later change triggers a targeted recheck when it alters the conditions behind an earlier decision.

Table 2: **Stage tasks and evidence.** Each stage starts from the retained bundle and may leave it unchanged if no candidate passes its checks.

Stage	Task and exit evidence
I. Prepare the streaming reference	
Reference contract	Fix weights, workload, precision, state lifecycle, and timing and behavior checks, then verify repeatable native rollout.
Repeated path	Separate reusable work from position, control, and history updates. Remove avoidable graph breaks and check first use, steady state, and request reset.
II. Compile and diagnose	
Compiled DiT	Compile the recurring DiT path and document fallback and remaining graph breaks.
Attention by site	Compare eligible backends by role, shape, mask, and compiled layout. Select using full-model results.
Post-compile profile	Identify recurring costs in the retained bundle to guide the next trials.
III. Reduce remaining DiT cost	
Residual work	Test redundant materialization or data movement, and operator or projection fusion, where the compiled path exposes them.
Graph replay	If launches remain costly, evaluate denoising and KV refresh as separate capture targets. Narrow from phase to blocks if needed, checking that replay uses the current state.
IV. Optimize native streaming VAE	
Streaming decode	After DiT, optimize native VAE compilation and operators. Compare sustained decoding with the parent while preserving temporal-cache updates.
Report	
Final report	Summarize retained and rejected changes, latency and behavior evidence, and remaining limitations.

Sequence parallelism (SP) or tensor parallelism (TP) can be explored afterward only if the user chooses a separate multi-GPU configuration.

A.1 AGENT CONFIGURATION AND EVALUATION SCOPE

The optimization runs underlying these case studies used GPT-5.6 Sol (model identifier `gpt-5.6-sol`) as the agent, with the `high` reasoning-effort setting. Final latency and video-quality measurements evaluate the retained implementations, not the agent's runtime or cost. During the optimization work, the agent inspected the target implementation, consulted stage-relevant technique records, proposed and implemented candidates, and ran the required probes and integrated checks. The harness supplied the stage order and evidence requirements without prescribing a winning configuration. Researchers fixed the workloads and reviewed ambiguous measurements and generated videos. Both Solaris and SolarWM-Wan5B informed workflow development, so these results are model-specific case studies rather than a held-out test of knowledge transfer. We did not compare agent models or effort settings, nor measure the reduction in human optimization time.

At each stage, the agent starts from the last retained bundle. A focused trial checks feasibility under the relevant shapes and state regime, but cannot by itself establish deployment benefit. A candidate considered for adoption is run with the retained optimizations and checked for steady-state performance and affected behavior. The trial record binds the parent bundle, candidate, workload,

measurement boundary, result, and decision. A failed or invalid trial leaves the parent unchanged. When a later change alters the prerequisites of an earlier decision, only that decision is reopened.

A.2 An attention experience card: from EFFICIENT to MATH

The Solaris action module illustrates why the technique library records both the selected backend and the conditions under which it performs well. A profile covering three latents recorded 225 action cross-attention calls: each of 15 attention blocks was invoked in four denoising forwards and one KV-refresh forward per latent. At this site, the K/V length was only four. With main self-attention and static cross-attention already using FlashAttention, the workflow examined whether another backend would better suit this small action-attention workload.

An earlier microbenchmark used BF16 queries of shape $(880, 16, 1, 64)$ and K/V length four. EFFICIENT, MATH, and FlashAttention took 313, 155, and 790 μs, respectively. These results made MATH a candidate for further evaluation. StreamAccel then re-evaluated self-attention, static cross-attention, and action cross-attention in the frozen compiled configuration. At the action site, the default EFFICIENT path took approximately 0.309 ms, compared with 0.061 ms for MATH. Tests at the other sites supported retaining their Flash paths.

To determine whether this operator-level advantage translated into a benefit for the existing optimization stack, StreamAccel switched only action attention to MATH and kept the other attention choices unchanged. It then measured the complete compiled model during sustained streaming generation. Under the tested configuration, DiT latency per latent decreased by 13.68%. Video-health and action-intervention checks also passed, so the change was retained.

The resulting experience card connects this decision to the attention site, computation shapes, mask, hardware, parallel layout, and compilation conditions, together with the results of local screening and full-model validation. For a new model, it guides the agent to compare eligible backends at the relevant sites and confirm gains within the current optimization stack. Changes to the Q/K/V layout, compilation, cache policy, or hardware therefore call for a fresh evaluation of the backend choice.

A.3 A graph experience card: capture boundaries and state updates

When profiling identifies substantial kernel-launch overhead, StreamAccel tests CUDA Graph replay for recurrent DiT computation. The capture boundary must reflect how each computation uses history: denoising reads the same history across multiple steps, whereas KV refresh writes new context once per chunk. StreamAccel therefore evaluates separate captures for these two phases, keeping the refresh update outside the graph replayed at every denoising step.

Within each phase, the agent starts with the largest safe capture region. If dynamic inputs or buffer lifetimes prevent reliable capture, it reduces the scope to groups of DiT blocks or a single block. Replay also needs the inputs for the current chunk: when positions and controls are held in fixed buffers, their contents must be updated before use. Together, these choices allow the captured computation to be reused while generation advances.

In the Solaris single-device evaluation, denoising and KV refresh were captured separately and replayed throughout sustained generation without recapture or fallback. The resulting experience card records the capture boundaries, required input updates, and continuation checks. For a new model, these records guide capture scope and input preparation. Steady-state timing and behavior checks then determine whether to retain the integrated path.

B Timing Protocol

The comparisons in Section 5.2 use completed GPU event intervals after warmup, with the profiler disabled. One-time compilation, capture, and cache filling are outside the steady-state window. In contrast, the measured intervals include the recurring work of denoising, necessary KV refresh, native streaming decode, and decoder-state maintenance within their respective component boundaries. Reference and StreamAccel perform matched work within each model. The reported reference is the mean of the two surrounding control legs. If their means are T_{A_1} and T_{A_2} and the candidate mean is T_B, the reported latency reduction is $100\,[1 - T_B/((T_{A_1} + T_{A_2})/2)]\,\%$. This bracketing

helps detect performance drift. Repeated chunk measurements within one rollout are correlated and are not independent runs of the optimization agent. The timing runs below are separate from the long-video quality cohorts in Appendix C. Table 3 specifies each model's native chunk size and the steady-state intervals included in the reported averages.

Table 3: **Steady-state timing windows.** The two models have different native chunk sizes, with equal work compared within each model.

Model	Unit	Measured window
Solaris	One mature latent (the native Solaris chunk) and its streaming decode, yielding four new frames	For each of three 257-latent runs, discard the first 16 latents and average the remaining 241 intervals.
SolarWM-Wan5B	One three-latent chunk and its three native VAE decodes, yielding twelve new frames	29 mature chunk positions $\{30 + 3k \mid k = 0, \ldots, 28\}$.

For Solaris, all three legs use the same test episode, start frame 204, seed 42, weights, BF16 precision, and four denoising steps on one H20. The archived measurement environment used PyTorch 2.8.0+cu128 and CUDA 12.8. For each chunk, the DiT interval includes denoising and the refresh needed for subsequent generation. The corresponding VAE interval covers one native streaming decode and its temporal-cache update. The two reference DiT measurements are 287.157 and 290.866 ms, giving a mean of 289.011 ms. StreamAccel reduces this to 178.220 ms. For VAE decoding, the reference measurements of 130.836 and 130.961 ms yield a mean of 130.898 ms, compared with 98.383 ms for StreamAccel.

For SolarWM-Wan5B, Reference and StreamAccel use the same archived `abot` case, starting at frame 123 with noise seed 1602803860. Both implementations receive identical camera controls and recorded noise tensors, and run with the same EMA weights, BF16 precision, and four denoising steps on one H20. These shared settings ensure a matched workload. We measure generation and decoding for the same chunk, which contains three latents. The DiT interval covers all four denoising forwards and the KV refresh required for subsequent generation. Its mean latency decreases from 1349.060 ms for Reference to 914.532 ms for StreamAccel. To measure decoding for the corresponding output, we time the three native VAE calls together with the accumulation between them. This directly measured interval decreases from 1280.712 to 1232.138 ms, rather than being estimated from the average duration of an individual decoder call.

C Quality and Control Evaluation

C.1 Matched rollouts and visual metrics

For each model, the final visual-quality cohort contains 20 source cases evaluated with three seeds each, yielding 60 Reference–StreamAccel pairs. The Solaris sources are drawn from eight test episodes. Within each pair, the two implementations use the same source and conditioning. We evaluate visual quality and generation behavior, not pixel identity.

To assess quality throughout each rollout, we sample aligned 128-frame windows from its early, middle, and late portions, as listed in Table 4. We compute the five VBench dimensions introduced in Section 5.1 for each window, then average the three window scores to obtain one score per video and dimension. The reported results are the means of these video-level scores across the cohort. This aggregation gives each video equal weight without treating its windows as independent cases. Imaging quality uses a 0–100 scale, while the other four dimensions use a 0–1 scale.

Table 4: **VBench windows.** Indices are zero-based, with the right endpoint excluded. The windows sample early, middle, and late sustained generation rather than only the initial frames.

Model	Early	Middle	Late
Solaris (1025 frames)	$[257, 385)$	$[513, 641)$	$[897, 1025)$
SolarWM-Wan5B (477 frames)	$[97, 225)$	$[225, 353)$	$[349, 477)$

C.2 Model-specific control measures

Solaris accepts discrete steering actions, so we measure control response through the correlation between steering inputs and scene motion. We encode the input as right-minus-left steering and estimate horizontal optical flow in the central scene, excluding the HUD and player-character regions. To account for response delay, we compare the two signals over temporal offsets from -8 to $+8$ latent steps and take the maximum sign-correct Pearson correlation for each video. Averaging these video-level scores yields 0.1670 for Reference and 0.1697 for StreamAccel.

SolarWM-Wan5B accepts camera controls, which we evaluate using WBench NavScore (Ying et al., 2026). WBench estimates camera poses from the generated video with MegaSaM and constructs a reference trajectory from the navigation instructions. After coordinate alignment and per-turn arc-length resampling, normalized translation and rotation errors determine navigation accuracy. A second component measures trajectory consistency across repeated or symmetric actions by comparing their motion in aligned local coordinate frames. We average each component over the cases where it is defined, then combine the two means with equal weight. The reported NavScore uses a 0–1 scale, with higher values indicating more accurate and consistent navigation. Reference and StreamAccel achieve NavScores of 0.8220 and 0.8222, respectively, indicating comparable camera-control performance after acceleration.

C.3 Additional qualitative comparisons

Figure 6 adds five matched SolarWM-Wan5B rollouts to Figure 5: an architectural passage, a snowy street, a courtyard, a forest landscape, and a hedge-lined walkway. Each pair shares its starting image, camera conditioning, weights, and noise. Both outputs retain a recognizable scene through the final frame, although details may differ. Across these examples, StreamAccel preserves the reference's overall visual quality without perceptible degradation.

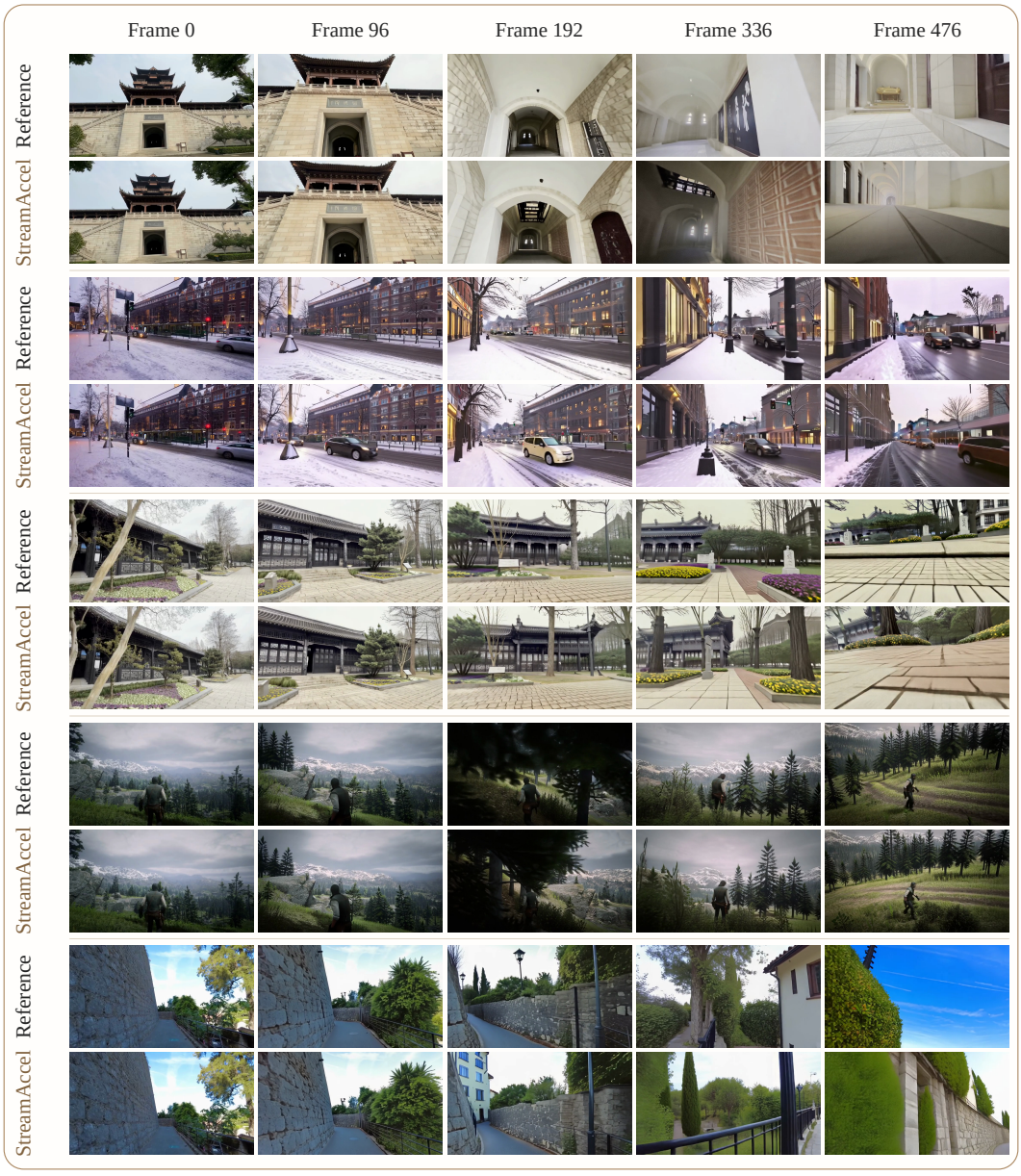

Figure 6: **Additional SolarWM-Wan5B visual comparisons.**