SOL VIDEO INFERENCE ENGINE: AGENT-NATIVE FULL-STACK ACCELERATION FRAMEWORK FOR EFFICIENT VIDEO GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern video diffusion models scale toward higher generation quality, but this scaling also increases inference cost. While a variety of acceleration methods have been proposed, one significant difficulty is that the most effective acceleration implementation is highly **instance-specific**: the best recipe for one *(model, hardware, inference configuration)* combination often fails to transfer to another. Models differ in architecture, numerical sensitivity, and attention concentration patterns. Inference configurations vary in spatial and temporal resolution as well as video duration, while hardware platforms differ in memory hierarchy, supported numerics, and kernel throughput. Together, these factors create a large tuning space in which manual performance engineering becomes costly. To address this, we present **Sol Video Inference Engine**, an agentic native training-free acceleration framework for video diffusion models. It organizes five broadly applicable techniques—cache, sparse attention, token pruning, quantization, and kernel fusion—into an agentic acceleration stack for instance-specific optimization. For a concrete *(model, hardware, serving configuration)* deployment target, parallel skill agents optimize the implementation of each technique, an agent integrator composes them into a global stack, and a human validator provides feedback on quality. We instantiate this workflow on four video models with different model sizes and architectures: 64B **Cosmos3-Super**, 33B **MiniMax-H3**, 22B **LTX-2.3**, and 2B **SANA-Video**. With little human effort, the full stack achieves more than $2\times$ end-to-end acceleration on Cosmos3-Super, LTX-2.3, and SANA-Video while maintaining near-lossless VBench quality, and $3.95\times$ on MiniMax-H3 with 8 GB200 GPUs ($3.55\times$–$4.52\times$ across six GPU platforms), demonstrating the effectiveness of the agent framework for video diffusion acceleration.

1 INTRODUCTION

Video diffusion models continue to scale toward higher generation quality, longer duration, and higher fidelity. This scaling, however, also increases inference cost. Modern video diffusion systems combine billion-scale backbones, long spatiotemporal sequences, and multi-step sampling, making inference computationally heavy.

While many acceleration methods have been proposed, accelerating these systems is not a one-time optimization problem. One significant difficulty is that effective acceleration is highly **instance-specific**: the best recipe for one *(model, hardware, inference serving configuration)* combination often fails to transfer to another. Models differ in architecture, numerical sensitivity, and attention concentration patterns. Inference configurations vary in spatial and temporal resolution, video duration, denoising schedule, and batching strategy, while target hardware ranges from high-memory datacenter cards to consumer-grade edge devices with distinct supported numerics and throughput.

As a direct result of these heterogeneous properties, the implementation of various acceleration methods is instance-dependent: a cache policy may be valid for one denoising schedule but drift under another; a sparsity pattern may preserve quality at one resolution but fail at a longer temporal

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.

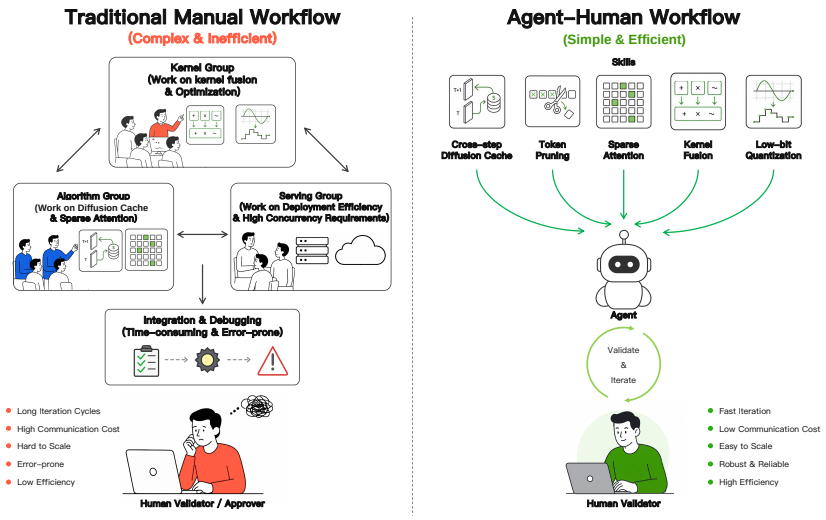
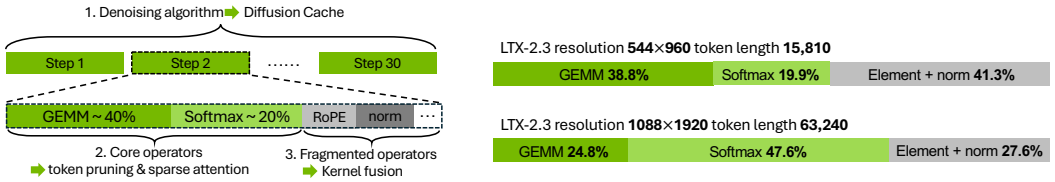


Figure 1: **Traditional manual acceleration workflow versus Sol Video Inference Engine's agent-human workflow.** Left: acceleration work is split across multiple human groups with different expert techniques, leading to long iteration cycles, high communication cost, and manual integration or debugging effort. Right: Sol Video Inference Engine exposes acceleration skills to an agent system that performs local search and stack integration, while the human stays in the loop mainly for validation and iteration. This agent-human mode reduces coordination overhead and human effort while making the workflow faster, easier to scale, and more reliable.

context; and a low-precision backend may provide a good speed–quality trade-off for one hardware but offer a poor trade-off for another. Because the combination space spanned by these factors is so large, hand-tuning every deployment instance consumes substantial engineering effort.

Sol Video Inference Engine addresses this challenge with an **agentic acceleration framework** organized around two coupled parts. The first is a full-stack set of acceleration techniques spanning cross-step cache, sparse attention, token pruning, quantization, and kernel fusion. The second is an agent-native workflow: parallel skill agents perform per-technique local optimization, an agent integrator composes the resulting candidates through global search, and a human validator provides visual-quality feedback. Sol Video Inference Engine applies this framework to 64B Cosmos3-Super, 33B MiniMax-H3, 22B LTX-2.3, and 2B SANA-Video, producing effective acceleration stacks with little manual effort. The contributions of this paper can be summarized in four aspects:

- **Problem reformulation.** We systematically analyze the acceleration space of video diffusion inference and formulate its acceleration as an instance-specific tuning problem rather than one-time optimization paradigms.
- **Full-stack acceleration framework.** We propose a comprehensive framework for video diffusion models acceleration, synergistically integrating algorithmic and system-level techniques, including cross-step caching, sparse attention, token pruning, quantization, and kernel fusion.
- **Agent-driven optimization workflow.** We propose an agent-native architecture that orchestrates parallel agents and an integrator. It successfully bypasses the multi-team engineering traditionally required for model deployment, autonomously delivering highly optimized speed–quality configurations in a fraction of the usual human effort.
- **Extensive empirical validation.** Extensive experiments demonstrate that our training-free framework achieves over $2\times$ end-to-end speedup across diverse state-of-the-art models—including 64B Cosmos3-Super, 22B LTX-2.3, and 2B SANA-Video—without compromising visual quality, and $3.95\times$ on the 33B MiniMax-H3 with 8 GB200 GPUs, with $3.55\times$–$4.52\times$ across six GPU platforms from A100 to DGX Spark.



(a) Three-level redundancy.

(b) Time profile across diffusion configurations under the same model and hardware.

Figure 2: **Baseline inference time breakdown and configuration-specific bottlenecks.** Left: runtime decomposition that illustrates algorithm-, model-, and kernel-level redundancy in the unaccelerated serving path. Right: time profile across diffusion configurations under the same model and hardware. Different spatial resolutions shift the dominant inference bottlenecks and therefore expose different primary optimization opportunities.

Relation to prior work. Existing acceleration methods for video diffusion, including caching, sparse attention, token pruning, quantization, and kernel fusion, each target one source of cost and are tuned by hand for a given setting; we use them as the building blocks of our stack (Sec. 3.2). Agentic systems such as CUDA-LLM and CudaForge generate or optimize individual CUDA kernels with compilation, correctness, and profiling feedback (Chen et al., 2025c; Zhang et al., 2025g). In contrast, our agents select, tune, and compose multiple acceleration techniques into a full serving stack under end-to-end quality constraints. Sec. A reviews related work in full.

2 Background: Inference Redundancy and Deployment Heterogeneity

2.1 Inference Redundancy in Video Diffusion Models

Scaling video diffusion models improves generation quality, temporal consistency, and controllability, but it also substantially increases inference cost. Recent systems illustrate this trend: Cosmos3-Super scales to 64B parameters for high-capacity video and physical-aware content generation (NVIDIA, 2026a), while LTX-2.3 is released as a 22B-parameter audio-video diffusion foundation model (Lightricks, 2026a). As parameter count and model scale increase, video diffusion inference also exposes redundancy at three complementary levels. At the **diffusion algorithm** level, adjacent denoising steps often execute structurally similar computations over slowly changing latent states, creating opportunities for cross-step reuse, selective skipping, and compensation. At the **model** level, long spatiotemporal sequences contain redundant tokens and attention interactions, so the full attention graph or full token set is not always necessary at every layer and timestep. At the **kernel** level, transformer inference repeatedly launches memory-bound operators around GEMMs, materializes intermediate tensors, and pays overhead for layout movement, normalization, activation, and precision conversion. Figure 2a visualizes how the three levels of redundancy appear in the unaccelerated inference path.

2.2 Deployment Heterogeneity: Model, Hardware, and Configuration

We treat an acceleration target as a heterogeneous deployment instance defined by three co-equal axes: *model, hardware, and inference serving configuration*. Acceleration implementations fail to transfer cleanly because each deployment instance is constrained by the joint choice of them, spanning a large combination space as shown in Figure 3. The **model** determines numerical and structural sensitivity: different backbones have different quantization robustness, attention redundancy, cache stability across denoising steps, and tolerance to token pruning. The **hardware** determines which bottlenecks are exposed: on high-compute GPUs such as B200, large GEMMs may become sufficiently fast that small-operator launch overhead and memory-bound epilogues occupy a larger fraction of runtime, while on lower-compute GPUs those same launches may be less visible because dense compute remains dominant. The **configuration** axis covers runtime choices such as spatial and temporal resolution (*e.g.*720p *vs.*1080p, 24fps *vs.*48 fps). It changes the effective token count and

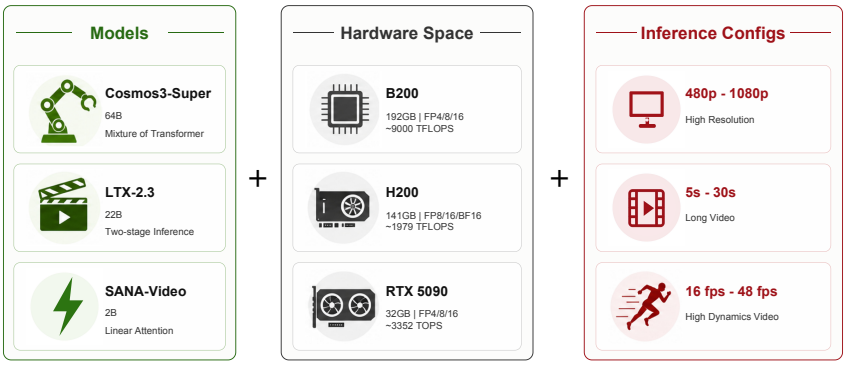


Figure 3: **Joint combination space of video generation model deployment instances.** Model, hardware, and inference configuration jointly define a large combinatorial space of deployment instances. This diversity makes implementation of acceleration techniques a substantial engineering challenge rather than a one-time optimization problem.

runtime balance: increasing spatial or temporal resolution increases sequence length, attention scales quadratically with token count, and FFN/MLP work scales roughly linearly, so the attention–FFN time split can change substantially across resolutions. These coupled constraints make caching, sparsity, pruning, fusion, and quantization properties instance-specific; an acceleration recipe that works for one *(model, hardware, serving configuration)* point may not preserve speed or quality at another. Figure 2b illustrates this effect within a single model and hardware setting: different resolutions expose different dominant bottlenecks and therefore different primary acceleration directions.

This creates two consequences for performance engineering. First, the number of deployment instances is large, so exhaustive manual tuning does not scale. Second, the optimization problem within each instance is structured: objectives such as latency and quality-preservation are explicit, which naturally lends itself to agent-based automation. Sol Video Inference Engine on Cosmos3-Super, LTX-2.3, and SANA-Video instantiates distinct points in this configuration space, demonstrating how an agent-native workflow can produce effective stacks for concrete deployment instances.

3 Agentic Acceleration Stack

Sol Video Inference Engine builds a full-stack acceleration framework that spans three levels of acceleration: algorithm-level cache to reduce redundancy across repeated denoising steps, model-level sparse attention and token pruning to reduce unnecessary token computation, and kernel-level optimizations through kernel fusion to minimize kernel-launch overhead, alongside quantization to accelerate execution by leveraging lower-precision hardware primitives. For a specific deployment instance, defined by a concrete *(model, hardware, serving configuration)* combination, the framework aims to select and compose the most suitable subset of these acceleration skills under acceptable generation quality. Within this framework, each acceleration technique is first tuned locally to identify the configuration that best fits the current deployment instance. The agent integrator then composes these per-technique optima and performs global tuning to find the strongest full-stack combination under generation-quality constraints. Sol Video Inference Engine delivers roughly $2\times$–$3\times$ end-to-end acceleration across Cosmos3-Super, LTX-2.3, and SANA-Video, which is achieved by a full-stack techniques composition rather than a single acceleration method, as shown in Figure 4.

3.1 Agent-Native Acceleration Workflow

Sol Video Inference Engine uses an **agent-native acceleration workflow** to tune and compose a deployment stack for each instance. The workflow turns the technique families detailed in Section 3.2 into a structured optimization process. Illustrated in Figure 5, it proceeds in three stages: parallel skill agents perform model-specific local optimization, an agent integrator composes the local candidates into a global stack, and a human validator provides quality feedback.

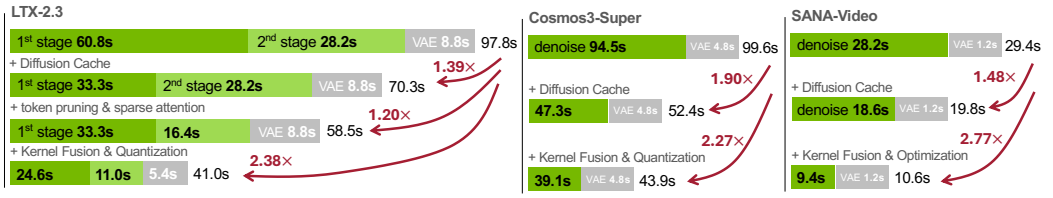


Figure 4: **Acceleration breakdown across different optimization methods.** The figure decomposes the cumulative speedup contributed by different components in the Sol Video Inference Engine stack, showing how cache, sparse attention and token pruning, and kernel-level optimization progressively reduce end-to-end latency. Overall, the full framework delivers roughly 2×–3× acceleration across Cosmos3-Super, LTX-2.3, and SANA-Video.

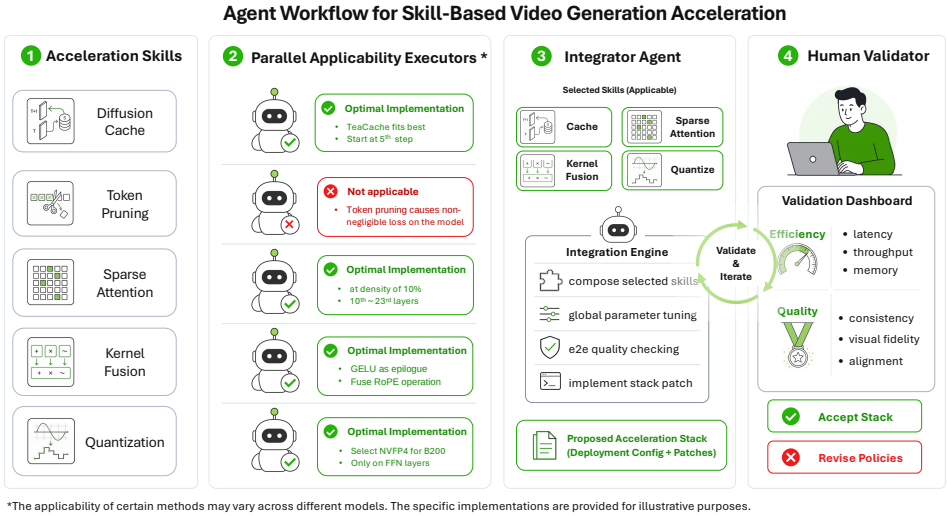


Figure 5: **Agent-native acceleration workflow.** The workflow can be viewed in two parts. The first part is local parallel per-technique search, where skill agents independently explore local candidates. The second part is global integration and the human-validator loop, where an integrator composes the local candidates into a full acceleration stack and the human validator provides feedback to guide the next iteration.

- **Parallel skill agents.** We first perform local optimization within each technique family. Because the full stack contains many techniques and the joint search space is large, directly running a single agent over the entire global optimization problem would be costly and slow. Instead, independent skill agents search their own local spaces in parallel. For example, a cache agent searches skip-step schedules and compensation rules, while a kernel-fusion agent profiles the kernel-wise time cost and tests fusions such as GEMM + GELU or RoPE + norm. These locally optimized candidates provide strong per-skill starting points for the later global optimization stage.
- **Agent as integrator.** The integrator receives local candidates from all skill agents and combines them into a full acceleration recipe for the target deployment. Because these methods are not independent, the final end-to-end speedup is not the direct product of their local gains. Moreover, many techniques are lossy, so directly stacking the locally optimal choices can accumulate approximation error and produce an unacceptable final result. The integrator therefore performs global optimization rather than simply aggregating first-stage results. It resolves interactions such as whether lossy methods like cache and sparse attention remain acceptable when combined, and whether quantization admits a compatible implementation together with kernel fusion under the same deployment target.
- **Human validator feedback.** Similarity metrics such as PSNR are not well aligned with human visual perception. Given a high-quality reference video, PSNR may assign a large distance

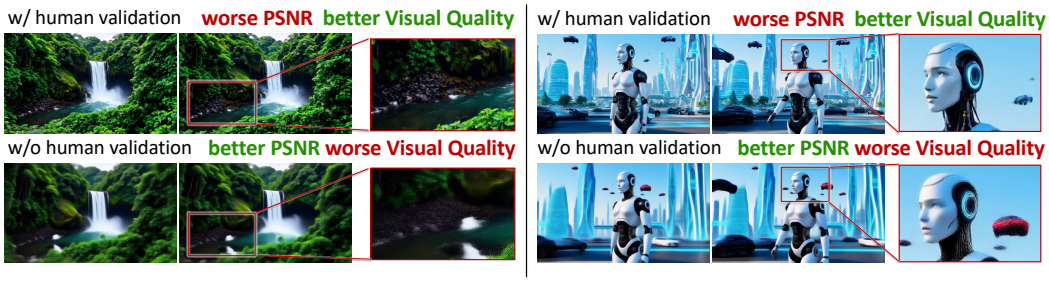


Figure 6: **Necessity of human validation.** Human validators inspect representative generations and provide feedback on overall visual quality, while traditional similarity operators such as PSNR often fail to faithfully reflect the actual perceptual quality of generated videos in deployment-oriented evaluation.

to outputs with shifted visual details, while assigning a smaller distance to outputs with blur, temporal jitter, or degraded motion coherence, as shown in Figure 6. From a human-quality perspective, the former is acceptable while the latter is not, which makes human feedback necessary. In the workflow, we use a fixed validation set to inspect each integrated candidate and determine whether its visual quality remains acceptable. This feedback then tells the next optimization round whether the global agent can push toward more aggressive acceleration or should stay more conservative to preserve quality.

It seeks strong local configurations for each technique family and composes them into a global operating point under quality constraints. In this process, parallel agents optimize each technique family locally, the integrator searches over the resulting candidate points, and the human validator verifies visual quality. Together, this workflow enables instance-specific optimization for each deployment target without requiring large manual tuning effort.

3.2 Acceleration Techniques

The skill agents draw on five technique families. For each family, we summarize the redundancy it targets and what the corresponding agent tunes; Sec. C gives the full descriptions and the related methods.

Cross-step cache. High-quality diffusion sampling runs tens of structurally similar denoising steps, and adjacent steps often produce similar intermediate features. Cache methods skip selected steps and reuse or forecast their outputs (Liu et al., 2024a; Zhou et al., 2025; Liu et al., 2025b). The cache agent selects the policy that matches the model and tunes its skip-step schedule, cached feature or residual, compensation rule, warmup, and maximum number of consecutive cached steps.

Sparse attention. Long spatiotemporal token sets are redundant across adjacent frames and within each frame, so many attention blocks contribute little to the output (Li et al., 2026b; Zhang et al., 2025d; Xi et al., 2025; Yang et al., 2025a). The sparse-attention agent selects the sparse backend, decides which layers can be sparsified and which must stay dense, and tunes the sparsity pattern and compensation policy.

Token pruning. Pruning shortens the sequence itself by skipping, merging, or reconstructing redundant latent tokens, which lowers the cost of all subsequent attention and feed-forward computation (Bolya & Hoffman, 2023; Liu et al., 2025a; Li et al., 2026c;e). The token-pruning agent tunes the pruning criterion, ratio, layer and timestep schedules, and reconstruction rule, and checks each candidate for temporal coherence and late-step fidelity.

Quantization. The speedup of low-precision execution depends on the hidden dimension, the number of tokens, and the hardware, while the quality impact depends on the model (Wu et al., 2024; Chen et al., 2025b; Li et al., 2025a; Zhang et al., 2025b). The quantization agent therefore profiles

Table 1: **End-to-end acceleration across model deployments on NVIDIA B200 GPUs.** The table compares baseline inference systems against the fully optimized Sol-Engine configuration across Cosmos3-Super, LTX-2.3, and SANA-Video. Speedups are reported relative to the SGLang baseline, while official latencies are shown for reference. The final optimized system delivers roughly 2×–3× end-to-end speedup.

Framework	Cosmos3-Super (64B, GPU=4)		LTX-2.3 (22B)		SANA-Video (2B)	
	Latency (s)	Speedup	Latency (s)	Speedup	Latency (s)	Speedup
Official	108.3	-	118.1	-	34.2	-
SGLang	99.6	**1.00**×	97.8	**1.00**×	29.4	**1.00**×
Sol-Engine	43.9	**2.27**×	41.0	**2.38**×	10.6	**2.77**×

layer sensitivity and tensor shapes, and assigns precision per layer and timestep instead of converting the model uniformly.

Kernel fusion. Around each GEMM, transformer blocks spend substantial time on memory-bound operators such as bias addition, residual updates, normalization, and activations. Fusing them into GEMM epilogues or single kernels avoids writing intermediates to HBM and launching separate kernels (NVIDIA, 2025; Zhai et al., 2023; Guo et al., 2026). The kernel-fusion agent profiles the partially accelerated stack and chooses which operator sequences to fuse and whether to use custom or compiler-generated kernels.

4 Experiments

Sol Video Inference Engine applies the agent-native workflow to multiple video diffusion models. Each deployment is a distinct *(model, hardware, inference serving configuration)* instance: the deployments share the same technique stack, but concrete implementations must be applied, optimized, and validated per target. We evaluate three representative video models across different model sizes and architectures—Cosmos3-Super (64B) (NVIDIA, 2026a), LTX-2.3 (22B) (Lightricks, 2026a), and SANA-Video (2B) (Chen et al., 2025a)—to demonstrate the generalizable acceleration capability of our framework.

4.1 Experimental Protocol

Deployment Hardware. All experiments run on NVIDIA B200 GPUs. We first deploy each model in an SGLang-based serving stack and then develop the acceleration components on top of this deployment. Cosmos3-Super uses four GPUs with sequence parallelism (SP), while LTX-2.3 and SANA-Video use one GPU.

Metrics. We use end-to-end latency as the primary metric. For quality preservation, we use VBench (Huang et al., 2024b) to evaluate visual quality and motion quality under matched prompts and serving configurations.

Models. We choose Cosmos3-Super (NVIDIA, 2026a), LTX-2.3 (Lightricks, 2026a), and SANA-Video (Chen et al., 2025a) to evaluate acceleration under our agentic acceleration framework. These models span different parameter scales, from 2B to 64B, and cover distinct inference pipelines and architectural designs: Cosmos3-Super uses a Mixture-of-Transformers (MoT) structure for physical video generation, LTX-2.3 emphasizes multi-resolution two-stage inference, and SANA-Video replaces quadratic attention with linear attention. Accelerating all three models demonstrates that the framework can generalize across foundation models with varied scales and architectural designs rather than overfitting to a single model family. Appendix D describes each model and its serving pipeline.

4.2 Efficiency Analysis

Across all three deployments, Sol Video Inference Engine achieves more than 2× end-to-end acceleration despite substantial differences in model architecture, parameter scale, and inference pipeline.

This suggests that the agentic acceleration framework is not tied to a single backbone, but can identify and compose effective optimizations for diverse video diffusion systems. Table 1 summarizes the resulting efficiency gains, with SGLang normalized to $1.00\times$ and official latencies shown for reference.

Cosmos3-Super. Cosmos3-Super starts from a 108.3 s official pipeline and a 99.6 s SGLang deployment. The diffusion-cache agent explores cache variants for the 64B MoT setting and selects a TeaCache-style policy (Liu et al., 2024a), reducing latency from 99.6 s to 52.4 s ($1.90\times$ over SGLang). Quantization is also selective across denoising time: using low precision in early steps can introduce larger structural deviation, while using low precision in the final refinement steps can create high-frequency visual artifacts. The agent therefore assigns precision by timestep and module sensitivity rather than applying one uniform format everywhere. Kernel-wise optimization reduces latency to 43.9 s, yielding cumulative $2.27\times$ speedup over SGLang and $2.47\times$ over the official pipeline.

LTX-2.3. LTX-2.3 starts from a 118.1 s official pipeline and a 97.8 s SGLang deployment. Diffusion cache removes redundancy across repeated denoising steps; because the high-quality LTX pipeline uses the `res_2s` sampler, cache schedules originally tuned for Euler-style samplers do not directly fit this trajectory well. In our framework, the agent discovers a fixed-step skip strategy that gives a better speed–quality operating point under the same inference NFE budget, reducing latency from 97.8 s to 70.3 s ($1.39\times$ over SGLang). Sparse attention and token pruning target the high-resolution stage, where the larger token count increases attention and DiT-forward cost; sparse attention reduces unnecessary bidirectional dense attention, while token pruning removes redundant token computation, bringing latency to 58.5 s ($1.20\times$ incremental and $1.67\times$ cumulative over SGLang). Kernel fusion then compresses memory-bound and fragmented operators: for example, GELU can be fused as a linear-layer epilogue to avoid a separate kernel launch. Along with NVFP4 quantization, kernel-wise optimization reduces the final latency to 41.0 s, yielding $2.38\times$ speedup over SGLang and $2.88\times$ over the official pipeline.

SANA-Video. SANA-Video starts from a 34.2 s official pipeline and a 29.4 s SGLang deployment. SANA-Video already incorporates efficiency-oriented model design, especially linear attention, so model-level acceleration has less room and we focus on the diffusion algorithm and kernel levels. The cache agent selects EasyCache (Zhou et al., 2025) as the best policy for this pipeline, reducing latency from 29.4 s to 19.8 s ($1.48\times$ over SGLang). At the kernel level, replacing high-precision linear-attention kernels with low-bit kernels gives an additional roughly $1.2\times$ kernel-level speedup with very small reconstruction loss (PSNR above 30 dB), while `torch.compile` performs operator fusion and graph construction to reduce kernel-launch overhead; together, these changes reduce end-to-end latency to 10.6 s, yielding $2.77\times$ speedup over SGLang and $3.23\times$ over the official pipeline.

4.3 Quality Evaluation

Table 2 compares the unaccelerated baseline against the resulting full-stack accelerated pipeline using detailed VBench dimensions. The reported results demonstrate that the accelerated stacks maintain average evaluation scores on par with the baseline, introducing only negligible variations in visual and motion quality. The evaluation results indicate that the observed latency gains stem from eliminating redundant or inefficient operations rather than sacrificing perceptually critical computations, thereby preserving prompt adherence and overall visual fidelity.

Figure 7 provides qualitative side-by-side video outputs under matched prompts and serving configurations. The accelerated outputs may exhibit minor differences in fine visual details, but the overall visual quality, motion quality, and physical fidelity remain strong. Across the shown cases, the accelerated videos preserve the basic scene layout, subject arrangement, and motion structure of the unaccelerated baseline, which is the key requirement for deploying the optimized stack in a near-lossless serving regime.

We further evaluate the stack on NVIDIA B300, where the higher NVFP4 throughput yields a larger acceleration advantage (Appendix F). We also apply the workflow to the newly released MiniMax-H3 and LTX-2.5 across six GPU platforms, reaching up to $4.52\times$ and $4.68\times$ end-to-end speedup, respectively (Appendix G).

Table 2: **VBench quality evaluation under full-stack acceleration.** The table compares the unaccelerated baseline against the fully optimized Sol Video Inference Engine configuration across the retained VBench dimensions. Δ reports the relative percentage change with respect to the baseline.

Configuration	**Average Score**	Subject consistency	Background consistency	Temporal flicker	Motion smoothness	Aesthetic quality	Imaging quality	Overall consistency
Cosmos3-Super (64B)								
Baseline	**0.7759**	0.9687	0.9301	0.9859	0.9923	0.6134	0.7276	0.2133
Sol-Engine	**0.7775**	0.9723	0.9382	0.9877	0.9935	0.6197	0.7178	0.2133
Δ	**+0.21%**	+0.37%	+0.87%	+0.18%	+0.12%	+1.03%	-1.35%	+0.00%
LTX-2.3 (22B)								
Baseline	**0.7646**	0.9010	0.9245	0.9675	0.9871	0.6234	0.7012	0.2474
Sol-Engine	**0.7605**	0.9006	0.9137	0.9704	0.9840	0.6104	0.7013	0.2429
Δ	**-0.54%**	-0.04%	-1.17%	+0.30%	-0.31%	-2.09%	+0.01%	-1.82%
SANA-Video (2B)								
Baseline	**0.7864**	0.9730	0.9648	0.9626	0.9843	0.6650	0.6892	0.2660
Sol-Engine	**0.7847**	0.9750	0.9654	0.9646	0.9842	0.6624	0.6779	0.2637
Δ	**-0.21%**	+0.21%	+0.06%	+0.21%	-0.01%	-0.39%	-1.64%	-0.86%

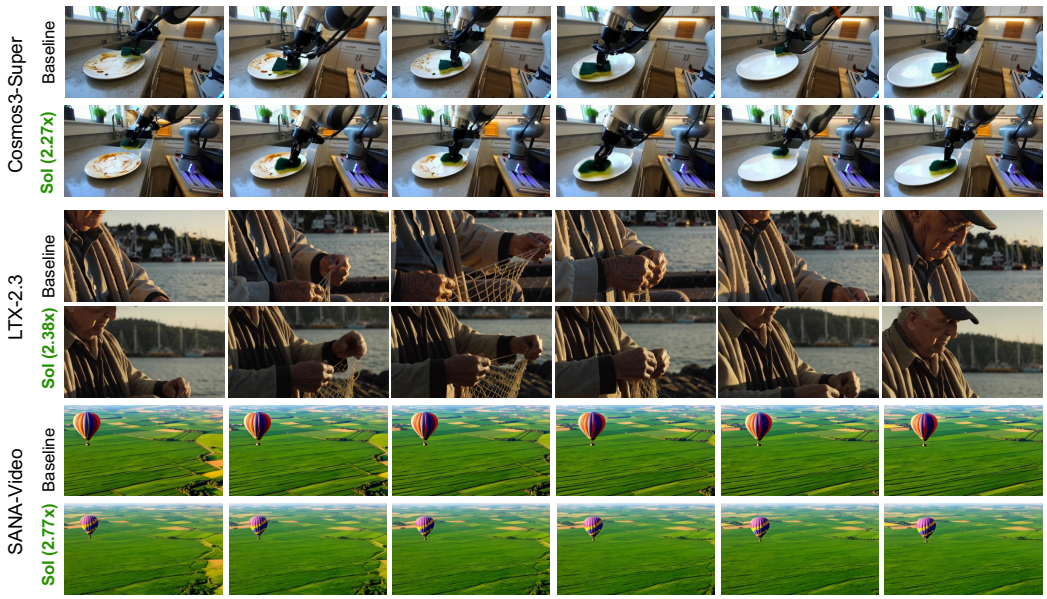


Figure 7: **Visual comparison on generated videos.** Qualitative demo comparing unaccelerated and accelerated outputs under matched prompts and serving configurations, with latency annotations. Our framework achieves substantial speedup without introducing visually perceptible quality differences.

5 Conclusion

Modern video diffusion models are rapidly improving in quality, but their inference cost remains a major obstacle to practical deployment, and effective acceleration is **instance-specific**: recipes tuned for one *(model, hardware, serving configuration)* combination often fail to transfer to another.

Sol Video Inference Engine addresses this gap with an **agentic acceleration stack** and an **agent-native acceleration workflow**. The stack organizes cache, quantization, efficient sparse attention, token pruning, and kernel optimization into model-specific design spaces; parallel skill agents tune local candidates, an agent integrator composes the selected candidates into a deployment stack, and a human validator provides feedback on the final efficiency–quality trade-off for targets including Cosmos3-Super, LTX-2.3, and SANA-Video. Across these models, it achieves more than $2\times$ end-to-end speedup while preserving visual and motion quality with negligible degradation. These results

show that agent-native full-stack acceleration can reduce inference cost with little human effort while largely maintaining generation quality. We discuss limitations and future work in Appendix B.

AI USE STATEMENT

Generative AI is a core component of the proposed method: coding agents implement, tune, and integrate the acceleration techniques studied in this paper, as described in Sec. 3. In addition, we used generative AI tools to implement methods and create or edit software code (the agent-produced acceleration components), and to create or modify scientific figures. Because the experiments are designed to evaluate the agentic workflow itself, coding agents lead the optimization process in each deployment. We also used large language models to polish the language and correct the grammar of the paper. Generative AI was not used to interpret the results. All agent-produced implementations were validated by human authors through end-to-end latency measurement, VBench evaluation, and visual inspection of generated videos. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The evaluated models, hardware platforms, serving stack, and evaluation protocol are described in Sec. 4, and the per-model configurations are listed in Sec. D. Additional qualitative results are provided in the appendix. We will release the code, configurations, and generated videos upon publication.

REFERENCES

Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. *CVPR Workshop on Efficient Deep Learning for Computer Vision*, 2023.

Junsong Chen, Yuyang Zhao, Jincheng Yu, Ruihang Chu, Junyu Chen, Shuai Yang, Xianbang Wang, Yicheng Pan, Daquan Zhou, Huan Ling, et al. SANA-Video: Efficient video generation with block linear diffusion transformer, 2025a. URL https://arxiv.org/abs/2509.24695.

Lei Chen, Yuan Meng, Chen Tang, Xinzhu Ma, Jingyan Jiang, Xin Wang, Zhi Wang, and Wenwu Zhu. Q-dit: Accurate post-training quantization for diffusion transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28306–28315, 2025b.

Wentao Chen, Jiace Zhu, Qi Fan, Yehan Ma, and An Zou. Cuda-llm: Llms can write efficient cuda kernels. *arXiv preprint arXiv:2506.09092*, 2025c.

Yukang Chen, Luozhou Wang, Wei Huang, Shuai Yang, Bohan Zhang, Yicheng Xiao, Ruihang Chu, Weian Mao, Qixin Hu, Shaoteng Liu, Yuyang Zhao, Huizi Mao, Ying-Cong Chen, Enze Xie, Xiaojuan Qi, and Song Han. LongLive-2.0: An nvfp4 parallel infrastructure for long video generation, 2026. URL https://arxiv.org/abs/2605.18739.

DefTruth, vipshop.com, etc. Cache-dit: A pytorch-native inference engine with cache, parallelism and quantization for diffusion transformers. https://github.com/vipshop/cache-dit.git, 2025. Open-source software. Accessed June 20, 2026.

Echo Team @ Joy Future Academy, JD. Joyai-echo: Pushing the frontier of long video generation. Technical report, Joy Future Academy, JD, May 2026. URL https://echo-team-joy-future-academy-jd.github.io/Echo-LongVideo-Page/. Project page. Accessed June 20, 2026.

Han Guo, Jack Zhang, Arjun Menon, Driss Guessous, Vijay Thakkar, Yoon Kim, and Tri Dao. Coda: Rewriting transformer blocks as gemm-epilogue programs. *arXiv preprint arXiv:2605.19269*, 2026.

Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. *arXiv preprint arXiv:2205.15868*, 2022.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. Mlagentbench: Evaluating language agents on machine learning experimentation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20271–20309, 2024a.

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*, pp. 21807–21818, 2024b.

Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. *arXiv preprint arXiv:2410.05954*, 2024.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

Haopeng Li, Yitong Li, Junsong Chen, Tian Ye, Haozhe Liu, Jincheng Yu, Duomin Wang, Ruihua Zhang, Zeke Xie, Enze Xie, and Song Han. Sol-Attn: Accelerating video generation inference via on-the-fly attention sparsification. *arXiv preprint arXiv:2607.24027*, 2026a.

Haopeng Li, Shitong Shao, Wenliang Zhong, Zikai Zhou, Lichen Bai, Hui Xiong, and Zeke Xie. Pisa: Piecewise sparse attention is wiser for efficient diffusion transformers. *arXiv preprint arXiv:2602.01077*, 2026b.

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. Svdquant: Absorbing outliers by low-rank component for 4-bit diffusion models. In *International Conference on Learning Representations*, 2025a.

Sheng Li, Yang Sui, Junhao Ran, Bo Yuan, Yue Dai, and Xulong Tang. Temporal aware pruning for efficient diffusion-based video generation. *arXiv preprint arXiv:2605.17837*, 2026c.

Xingyang Li, Muyang Li, Tianle Cai, Haocheng Xi, Shuo Yang, Yujun Lin, Lvmin Zhang, Songlin Yang, Jinbo Hu, Kelly Peng, Maneesh Agrawala, Ion Stoica, Kurt Keutzer, and Song Han. Radial attention: $\mathcal{O}(n \log n)$ sparse attention with energy decay for long video generation. *arXiv preprint arXiv:2506.19852*, 2025b.

Yitong Li, Junsong Chen, Shuchen Xue, Pengcuo Zeren, Siyuan Fu, Dinghao Yang, Yangyang Tang, Junjie Bai, Ping Luo, Song Han, and Enze Xie. Fp4 explore, bf16 train: Diffusion reinforcement learning via efficient rollout scaling, 2026d. URL https://arxiv.org/abs/2604.06916.

Zhuojin Li, Hsin-Pai Cheng, Hong Cai, Shizhong Han, and Fatih Porikli. Coredit: Spatial coherence-guided token pruning and reconstruction for efficient diffusion transformers. *arXiv preprint arXiv:2605.14191*, 2026e.

Lightricks. LTX-2.3 Model Card. https://huggingface.co/Lightricks/LTX-2.3, 2026a. Model checkpoint family including ltx-2.3-22b-dev and distilled variants. Accessed June 20, 2026.

Lightricks. LTX-2.5 Model Card. https://huggingface.co/Lightricks/LTX-2.5, 2026b. Accessed September 26, 2026.

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. *arXiv preprint arXiv:2411.19108*, 2024a.

Haosong Liu, Yuge Cheng, Wenxuan Miao, Zihan Liu, Aiyue Chen, Jing Lin, Yiwu Yao, Chen Chen, Jingwen Leng, Yu Feng, and Minyi Guo. Astraea: A token-wise acceleration framework for video diffusion transformers. *arXiv preprint arXiv:2506.05096*, 2025a.

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)*, pp. 15853–15863, October 2025b.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, 2024b.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

MiniMax. MiniMax-H3 Model Card. https://huggingface.co/MiniMaxAI/MiniMax-H3, 2026. Accessed September 26, 2026.

NVIDIA. Cutlass epilogue operations. https://nvidia-cutlass-22.mintlify.app/cpp/epilogue, 2025. Documentation. Accessed June 20, 2026.

NVIDIA. Cosmos 3: Omnimodal world models for physical ai. *arXiv preprint arXiv:2606.02800*, 2026a. URL https://arxiv.org/abs/2606.02800.

NVIDIA. Components — nvidia hgx ai factory. https://docs.nvidia.com/enterprise-reference-architectures/hgx-ai-factory/latest/components.html, 2026b. Accessed June 20, 2026.

Xuan Shen, Chenxia Han, Yufa Zhou, Yanyue Xie, Yifan Gong, Quanyi Wang, Yiwei Wang, Yanzhi Wang, Pu Zhao, and Jiuxiang Gu. Draftattention: Fast video diffusion via low-resolution attention guidance. *arXiv preprint arXiv:2505.14708*, 2025.

Meituan LongCat Team, Xunliang Cai, Qilong Huang, Zhuoliang Kang, Hongyu Li, Shijun Liang, Liya Ma, Siyu Ren, Xiaoming Wei, Rixu Xie, and Tong Zhang. Longcat-video technical report, 2025. URL https://arxiv.org/abs/2510.22200.

Tencent Hunyuan Foundation Model Team. Hunyuanvideo 1.5 technical report, 2025. URL https://arxiv.org/abs/2511.18870.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*, 2024.

Junyi Wu, Haoxuan Wang, Yuzhang Shang, Mubarak Shah, and Yan Yan. Ptq4dit: Post-training quantization for diffusion transformers. In *Advances in Neural Information Processing Systems*, 2024.

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, et al. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity. *arXiv preprint arXiv:2502.01776*, 2025.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying llm-based software engineering agents. *arXiv preprint arXiv:2407.01489*, 2024.

Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. Xattention: Block sparse attention with antidiagonal scoring. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R. Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024.

Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Chenfeng Xu, Kelly Peng, et al. Sparse videogen2: Accelerate video generation with sparse attention via semantic-aware permutation. *arXiv preprint arXiv:2505.18875*, 2025a.

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 *International Conference on Learning Representations*, 2025b.

Yujia Zhai, Chengquan Jiang, Leyuan Wang, Xiaoying Jia, Shang Zhang, Zizhong Chen, Xin Liu, and Yibo Zhu. Bytetransformer: A high-performance transformer boosted for variable-length inputs. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 344–355, 2023.

Jintao Zhang, Haofeng Huang, Pengle Zhang, Jia Wei, Jun Zhu, and Jianfei Chen. Sageattention2: Efficient attention with thorough outlier smoothing and per-thread int4 quantization. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 75097–75119, 2025a.

Jintao Zhang, Jia Wei, Haofeng Huang, Pengle Zhang, Jun Zhu, and Jianfei Chen. Sageattention: Accurate 8-bit attention for plug-and-play inference acceleration. In *International Conference on Learning Representations*, 2025b.

Jintao Zhang, Jia Wei, Haoxu Wang, Pengle Zhang, Xiaoming Xu, Haofeng Huang, Kai Jiang, Jun Zhu, and Jianfei Chen. Sageattention3: Microscaling fp4 attention for inference and an exploration of 8-bit training. *arXiv preprint arXiv:2505.11594*, 2025c.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattn: Accurate sparse attention accelerating any model inference. In *International Conference on Machine Learning (ICML)*, 2025d.

Peiyuan Zhang, Yongqi Chen, Haofeng Huang, Will Lin, Zhengzhong Liu, Ion Stoica, Eric P. Xing, and Hao Zhang. Faster video diffusion with trainable sparse attention. In *Advances in Neural Information Processing Systems*, 2025e.

Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhengzhong Liu, and Hao Zhang. Fast video generation with sliding tile attention. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 74714–74731, 2025f. URL https://proceedings.mlr.press/v267/zhang25m.html.

Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. Autocoderover: Autonomous program improvement. *arXiv preprint arXiv:2404.05427*, 2024.

Zijian Zhang, Rong Wang, Shiyang Li, Yuebo Luo, Mingyi Hong, and Caiwen Ding. Cudaforge: An agent framework with hardware feedback for cuda kernel optimization. *arXiv preprint arXiv:2511.01884*, 2025g.

Tianchen Zhao, Tongcheng Fang, Haofeng Huang, Rui Wan, Widyadewi Soedarmadji, Enshu Liu, 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 *International Conference on Learning Representations*, 2025.

Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. *arXiv preprint arXiv:2408.12588*, 2024.

Hailin Zhong and Shengxin Zhu. Ai harness engineering: A runtime substrate for foundation-model software agents. *arXiv preprint arXiv:2605.13357*, 2026.

Xin Zhou, Dingkang Liang, Kaijin Chen, Tianrui Feng, Xiwu Chen, Hongkai Lin, Yikang Ding, Feiyang Tan, Hengshuang Zhao, and Xiang Bai. Less is enough: Training-free video diffusion acceleration via runtime-adaptive caching. *arXiv preprint arXiv:2507.02860*, 2025.

A Related Work

A.1 Video Diffusion Models

Video generation is currently dominated by large-scale Diffusion Transformers (DiTs). Mainstream open-source models, such as CogVideo and CogVideoX (Hong et al., 2022; Yang et al., 2025b), the Wan series (Wan et al., 2025), HunyuanVideo (Kong et al., 2024; Team, 2025), and Cosmos3-Super (NVIDIA, 2026a), establish the standard paradigm for high-fidelity and physically-aware synthesis. To handle extended temporal contexts and high resolutions, systems like LongCat-Video (Team et al., 2025), LTX-2.3 (Lightricks, 2026a), JoyAI-Echo (Echo Team @ Joy Future Academy, JD, 2026), and Pyramid Flow (Jin et al., 2024) adopt multi-stage, autoregressive, or memory-augmented generation pipelines. Other models explore the extremes of the design space for efficiency: SANA-Video (Chen et al., 2025a) prioritizes efficiency through linear attention. Despite this architectural diversity, all these systems face a severe, shared inference bottleneck. The combination of massive spatiotemporal token sequences and iterative denoising steps inherently makes efficient deployment a critical system-level challenge.

A.2 Video Generation Acceleration

Prior acceleration work attacks video diffusion cost from three complementary levels. At the **algorithm** level, cache-based methods exploit similarity across denoising steps: TeaCache predicts timestep-to-timestep output change to decide when cached residuals can be reused (Liu et al., 2024a), and TaylorSeer forecasts features across timesteps using Taylor expansion to reduce the error of more aggressive reuse (Liu et al., 2025b). A broader spectrum of caching strategies has also emerged to accelerate diffusion inference (Zhou et al., 2025; DefTruth, vipshop.com, etc., 2025; Zhao et al., 2024). At the **model** level, sparse attention and token reduction reduce work inside each denoising step: PISA uses training-free piecewise sparse attention with first-order Taylor compensation (Li et al., 2026b), Sparse VideoGen and Sparse VideoGen2 exploit spatial-temporal sparsity with profiling and layout-aware sparse execution (Xi et al., 2025; Yang et al., 2025a), VSA learns video sparse attention operators (Zhang et al., 2025e). Many other methods have also extensively explored sparse attention for video diffusion (Zhang et al., 2025d; Shen et al., 2025; Zhang et al., 2025f; Xu et al., 2025; Li et al., 2025b; Chen et al., 2026). Meanwhile, token-reduction methods such as ToMe-SD, Astraea, TAPE, and CoReDiT merge, prune, or reconstruct redundant diffusion tokens (Bolya & Hoffman, 2023; Liu et al., 2025a; Li et al., 2026c;e). At the **kernel** level, quantization and fusion reduce the cost of the remaining operators: PTQ4DiT, Q-DiT, ViDiT-Q, and SVDQuant study diffusion-aware low-precision execution (Wu et al., 2024; Chen et al., 2025b; Zhao et al., 2025; Li et al., 2025a; 2026d). Moreover, CUTLASS epilogues, ByteTransformer, and CODA show how GEMM-adjacent memory-bound operations can be fused into high-throughput kernels (NVIDIA, 2025; Zhai et al., 2023; Guo et al., 2026). Sol Video Inference Engine builds on these technique families.

A.3 Agentic Workflows and Harness Engineering

Recent work increasingly treats the surrounding agent runtime—tool access, execution loop, verification, memory, and feedback—as a first-class optimization target rather than incidental glue code. AgentBench and MLAgentBench provide broad evaluation settings for language agents in interactive environments and machine-learning experimentation, respectively (Liu et al., 2024b; Huang et al., 2024a). In software engineering, SWE-agent, AutoCodeRover, Agentless, and OpenHands study how agent interfaces, search, localization, execution environments, and generalist developer agents affect autonomous code improvement workflows (Yang et al., 2024; Zhang et al., 2024; Xia et al., 2024; Wang et al., 2024). AI Harness Engineering formalizes the harness itself as a runtime substrate for foundation-model software agents (Zhong & Zhu, 2026), while The AI Scientist shows that agentic systems can orchestrate idea generation, experiment execution, figure production, and manuscript writing in an end-to-end loop (Lu et al., 2024). Closer to systems optimization, CUDA-LLM and CudaForge demonstrate that iterative agentic workflows with compilation checks, correctness validation, profiling, and hardware feedback can automatically generate or optimize efficient CUDA kernels (Chen et al., 2025c; Zhang et al., 2025g). Sol Video Inference Engine is aligned with this direction, but targets a different object of coordinating multiple acceleration skills for video diffusion inference into a quality-constrained full-stack deployment.

B Limitations and Future Work

While Sol Video Inference Engine demonstrates consistent acceleration across models with different scales, architectures, and serving pipelines, its current workflow still depends on human judgment for final quality assessment. Traditional automatic loss or similarity operators, such as PSNR-style measurements, are not well aligned with the visual intent of video generation. When an accelerated output changes harmless visual details, these metrics may report a large difference even though the result is acceptable to a human reviewer. Conversely, high-frequency noise, blur, temporal jitter, or physically implausible motion may remain numerically close to the baseline while being clearly unacceptable perceptually. Because these traditional detectors do not faithfully reflect human aesthetic intent and deployment expectations, the current system keeps human feedback validation in the loop: humans inspect representative generations, update the acceptable quality envelope, and redirect the agent search when automatic scores fail to capture visible artifacts.

This limitation also points to a natural direction for future work. A stronger end-to-end visual-quality evaluator could reduce the amount of manual validation, make the feedback loop more scalable, and allow agents to optimize more aggressively while staying within quality constraints. Future versions of the framework could incorporate learned video preference models, artifact detectors, physics-consistency checks, and VLM-based end-to-end quality judgment so that the integrator receives richer feedback than latency and aggregate benchmark scores alone. Improving automatic quality recognition would make the same agent-native acceleration workflow more autonomous and more broadly deployable.

C Acceleration Techniques in Detail

C.1 Cross-Step Cache

Diffusion inference typically requires a large number of function evaluations (NFE), often on the order of 30–50 denoising steps for high-quality generation. Across these steps, the model repeatedly executes structurally similar computation, and adjacent denoising states often induce similar intermediate features. Cross-step caching targets this trajectory-level redundancy by skipping selected denoising-step computations and compensating for the skipped output.

Recent diffusion-cache methods illustrate the range of usable cache policies. TeaCache estimates timestep-to-timestep output change from timestep-conditioned inputs and reuses cached residuals when the predicted change is small (Liu et al., 2024a). EasyCache introduces a runtime-adaptive caching policy that adjusts feature reuse online to better balance acceleration and quality across video diffusion trajectories (Zhou et al., 2025). TaylorSeer extends feature reuse into feature forecasting, using Taylor-series extrapolation over previous timestep features to reduce quality degradation when cached intervals become larger (Liu et al., 2025b). For a specific model and deployment configuration, Sol Video Inference Engine treats cache selection as an instance-specific optimization problem rather than a generalizable methodology. The cache agent must determine which policy best matches the model, then tune its key hyperparameters, such as skip-step schedule, cached feature or residual choice, compensation rule, warmup, and maximum cached-step constraints.

C.2 Sparse Attention

Video diffusion inference operates over large spatiotemporal token sets, especially for long-duration or high-resolution generation. These tokens exhibit both temporal redundancy across adjacent frames and spatial redundancy within each frame, so many attention edges contribute little to the final output on a given step. Sparse attention targets the quadratic cost of full spatiotemporal attention without necessarily changing the external token lattice.

Recent work shows several ways to exploit this sparsity. PISA provides training-free piecewise sparse attention: critical blocks are computed exactly, while non-critical blocks are compensated with first-order Taylor approximation (Li et al., 2026b). SpargeAttention introduces a universal training-free sparse attention framework that accurately predicts and skips near-zero attention entries using a block-wise similarity metric (Zhang et al., 2025d). Furthermore, Sparse VideoGen exploits spatial-temporal sparsity patterns with online profiling and custom kernels (Xi et al., 2025), while

Sparse VideoGen2 improves sparse-token identification and GPU layout through semantic-aware permutation (Yang et al., 2025a). For a specific model and deployment configuration, Sol Video Inference Engine must determine which sparse attention backend fits best in the setting, as well as which attention layers can be sparsified safely and which should remain dense for quality preservation. The sparse-attention agent therefore tunes layer selection, sparsity pattern, and compensation policy, then validates the quality–speedup trade–off under given inference configuration.

C.3 Token Pruning

Token pruning reduces the sequence itself by skipping, merging, or reconstructing redundant latent tokens. By reducing the number of active tokens, it lowers the cost of subsequent attention and feed-forward computation, especially in high-resolution or long-video settings where sequence length dominates latency.

Earlier token-reduction work such as ToMe for Stable Diffusion shows that redundant diffusion tokens can be merged at inference time with limited quality loss (Bolya & Hoffman, 2023). More recent video-diffusion systems make the pruning decision more temporally and spatially aware: Astraea searches token budgets for video DiTs under a performance target (Liu et al., 2025a), TAPE smooths token importance across frames to avoid temporal jitter (Li et al., 2026c), and CoReDiT prunes spatially coherent DiT tokens while reconstructing skipped outputs to preserve a dense representation (Li et al., 2026e). For a specific model and deployment configuration, the key question is not whether token pruning helps in principle, but which pruning policy removes redundant computation without destabilizing motion or fine detail. The token-pruning agent therefore tunes the pruning criterion, pruning ratio, layer schedule, timestep schedule, and reconstruction rule, then validates each candidate against temporal coherence and late-step fidelity before exposing it to stack integration.

C.4 Quantization

While quantization is typically employed to reduce computational cost, its actual speedup is heavily governed specifically by the model's hidden dimension, the number of inference tokens and the hardware characteristics. Concurrently, a model's sensitivity to quantization noise is fundamentally dictated by its inherent training parameter distribution. Because both the acceleration potential and the resulting quality impact vary drastically across various deployment instances, we must conduct an instance-specific search to identify the optimal quality–speedup trade-off.

Recent work reduces quantization loss through diffusion-aware calibration and scaling. PTQ4DiT identifies salient channels and timestep-varying activations as key post-training quantization challenges (Wu et al., 2024); Q-DiT uses automatic quantization granularity allocation for DiTs (Chen et al., 2025b); and SVDQuant absorbs weight and activation outliers with low-rank components to enable 4-bit diffusion inference (Li et al., 2025a). The SageAttention series targets the attention quantization, introducing 8-bit to 4-bit attention acceleration with outlier smoothing (Zhang et al., 2025b;a;c). For a specific deployment instance, Sol Video Inference Engine treats quantization as a selective local-tuning problem rather than a uniform low-bit conversion. The quantization agent profiles layer sensitivity and tensor shapes, then tunes layer-wise precision assignments, weight and activation bitwidths, timestep-dependent scaling to find the best speed–quality trade-off.

C.5 Kernel Fusion

Transformer and diffusion backbones spend substantial time around GEMMs on fragmented operators that have low arithmetic intensity: bias addition, residual updates, normalization, activation functions, scaling, and layout conversion. Kernel fusion accelerates these paths by executing memory-bound follow-up operators while intermediate results are still in registers or shared memory, instead of materializing them to HBM and launching separate kernels.

This pattern is explicit in CUTLASS epilogue programming, where activations such as GELU can be fused as GEMM epilogues (NVIDIA, 2025), and in ByteTransformer, which uses customized fused CUTLASS epilogues for bias and GELU to hide memory latency inside GEMM execution (Zhai et al., 2023). CODA generalizes this idea by rewriting transformer blocks as GEMM-plus-epilogue programs, fusing normalization, activations, residual updates, reductions, and related memory-

bound work into tile-local epilogue code (Guo et al., 2026). For a specific model and deployment configuration, Sol Video Inference Engine treats kernel fusion as a late-stage local-tuning problem driven by the current bottlenecks of the partially accelerated stack. The kernel-fusion agent profiles the kernel-wise time consumption, then tunes which operator sequences to fuse, whether to use custom kernels or compiler-generated fusion, and how to match the implementation to the current tensor shapes and precision formats. Typical candidates include GEMM + GELU, GEMM + residual, and fused RoPE + norm.

D Implementation Details

D.1 Overall Implementation Details

Across all three models, we vary only their concrete implementation scope and hyperparameters: cache policy, sparse attention replacement, token-pruning schedule, low-precision quantization scope, and kernel-level fusion. The deployment interface is always organized into a dense `baseline` path and a composed `fullopt` path, so the measured comparisons reflect end-to-end serving stacks rather than isolated kernel microbenchmarks. The runtime itself remains SGLang-based, while the optimized path selectively swaps in cache reuse, sparse or reduced token computation, low-precision kernels, and fused operator sequences according to each model's architecture and bottlenecks.

At a method level, the global implementation principles are consistent. Cache is always applied as a training-free denoising-step reuse policy, but the exact schedule differs by model. Sparse attention is only introduced when the attention map exhibits enough structure to justify replacing dense bidirectional attention with a piecewise sparse pattern. Token pruning is used only when intermediate video tokens contain enough redundancy that dropping a subset can reduce cost without visible artifacts. Quantization is applied selectively instead of uniformly: numerically fragile stages stay in higher precision, while stable GEMM-heavy regions are moved to low precision. Kernel optimization is similarly selective and includes epilogue fusion, QKV-path fusion, normalization-path fusion, and compiler-driven graph fusion when those transformations are stable for the target workload.

D.2 Model-Specific Pipelines

- **Cosmos3-Super.** Cosmos3-Super is a 64B-parameter large video model focused on physical generation (NVIDIA, 2026a). It uses a Mixture-of-Transformers (MoT) architecture with an autoregressive reasoning tower and a diffusion generation tower, scaling the model to 64B parameters and improving generation quality and physical reasoning capability.
- **LTX-2.3.** LTX-2.3 is a 22B-parameter audio-video diffusion model (Lightricks, 2026a). Its serving pipeline includes two stages: stage 1 generates the video latent at 544×960 with 15 denoising steps, stage 2 upsamples and refines at 1088×1920 with 3 denoising steps, followed by VAE decode for 241 frames. The high-quality pipeline uses the `res_2s` second-order sampler, which improves quality/step trade-offs but differs from the Euler-style schedules assumed by many cache heuristics, making cache implementation and optimization less direct.
- **SANA-Video.** SANA-Video is a 2B-parameter video diffusion model based on a block linear diffusion transformer (Chen et al., 2025a). It uses linear attention to reduce the $O(n^2)$ overhead of standard attention and convolution-enhanced FFN blocks to preserve local spatiotemporal structure, achieving strong video quality while remaining efficient.

D.2.1 Cosmos3-Super

Cosmos3-Super is dominated by very large transformer blocks, so the optimized pipeline focuses on reducing repeated denoising computation and accelerating the GEMM-heavy middle part of the trajectory. The cache component follows a TeaCache-style residual replay policy with threshold 1.15, start step 10, and at most three continuous cache hits. It measures the relative L_1 change of the generation hidden state across denoising steps, accumulates this change, and skips the generation-path transformer blocks only when the accumulated change remains below the threshold; otherwise it recomputes and refreshes the cached residual. Quantization is applied in a step-selective manner rather than uniformly. The first three and last three denoising steps are kept in the dense/high-precision path, because early steps determine global structure while late steps refine high-frequency details.

NVFP4 is then applied to the more regular middle steps and GEMM-heavy generation-path linear layers with high arithmetic density, including the FFN gate-up and down projections as well as the attention QKV and output projections. This combination makes Cosmos3-Super the case where cache and NVFP4-oriented transformer execution provide the clearest end-to-end gain.

D.2.2 LTX-2.3

LTX-2.3 uses a two-stage high-quality pipeline. Stage 1 runs the `res_2s` sampler for 15 denoising steps at 544×960, and stage 2 upsamples and refines the latent at 1088×1920 for 3 denoising steps before VAE decoding 241 frames. The full optimized stack uses a fixed-step cache method in stage 1 rather than an online TeaCache threshold. The concrete preset is `8of15_last_29calls`: over the 29 stage-1 denoiser calls, it reuses the last computed result at call indices [13, 14, 16, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27], with `reuse_mode=last` and `max_skip_steps=30`. Stage 2 applies PISA only to the high-resolution refinement transformer: sparsity is 0.9 (density 0.1), block size is 64, only video self-attention is replaced, stage-1 sparse scheduling is disabled, and stage-2 layers 0–1 remain dense while later layers use piecewise attention with approximate remainder enabled, score-based routing, and FlashAttention dense fallback. Token pruning is also restricted to stage 2: during the additional refinement model calls, the system keeps 50% of video tokens according to the `feat_norm` saliency score, and applies pruning on refinement calls 1 and 2. NVFP4 is applied only to the video FFN input and output projections; the NVFP4 fused `proj_in+GELU` and `proj_out+bias+gate` paths are disabled. The lossless kernel/fusion flags enabled in the full stack are block-0 self-attention sharing, guidance-prefix sharing, fused QK+RoPE, fused RMS-AdaLN, fused AdaLN, fused QKNorm+RoPE, fused dual modulation, fused cross-attention dual modulation, fused Ada values, fused residual gate, fused FFN `proj_in+GELU`, compiled gate-to-output, fused audio QKVG, global fused QKNorm+RoPE, and compiled tiled VAE decoding. The distilled LoRA strengths are 0.25 in stage 1 and 0.5 in stage 2; stage 2 uses sigmas $[0.909375, 0.725, 0.421875, 0.0]$.

D.2.3 SANA-VIDEO

SANA-Video uses a 2B linear-attention backbone and is evaluated with 81 frames, 50 denoising steps, and the 480p setting 832×480 in the main demo configuration. Its optimized path does not use sparse attention, token pruning, or NVFP4 FFN quantization. The cache component is EasyCache with threshold 0.1, warmup 3 steps, and spatial subsample stride 8. The decision rule estimates the online input-to-output transformation rate from subsampled hidden states, accumulates the estimated relative output change, and skips the transformer block stack while the accumulated value stays below the threshold; the cached residual is shared across the unbatched CFG branches to avoid injecting stale guidance. The remaining acceleration comes from three lossless or near-lossless kernel-path changes: the linear-attention key/value aggregation is kept in BF16 tensor-core execution instead of being promoted to FP32, self-attention Q/K/V projections are merged into one GEMM by concatenating the projection weights after loading, and the DiT block stack is compiled with torch.compile. The max-autotune compile mode uses `max-autotune-no-cudagraphs`, subprocess autotuning, and a persistent TorchInductor cache; the safe compile path uses the default torch.compile mode. The 480p setting reports roughly 16 skipped steps out of 50 for EasyCache, with the remaining speedup coming from BF16 linear attention, QKV merge, and compile/fusion.

E VISUAL DEMO COMPARISON

We provide extended visual comparisons to show that the Sol-Engine accelerated pipelines remain visually close to the unaccelerated SGLang deployments while achieving more than $2\times$ end-to-end speedup. For each of Cosmos3-Super, LTX-2.3, and SANA-Video, we collect six matched video pairs. In each figure, the upper strip shows the unaccelerated SGLang result and the lower strip shows the Sol-Engine result for the same prompt, making it possible to inspect whether scene layout, motion evolution, and visual fidelity are preserved under the accelerated setting.

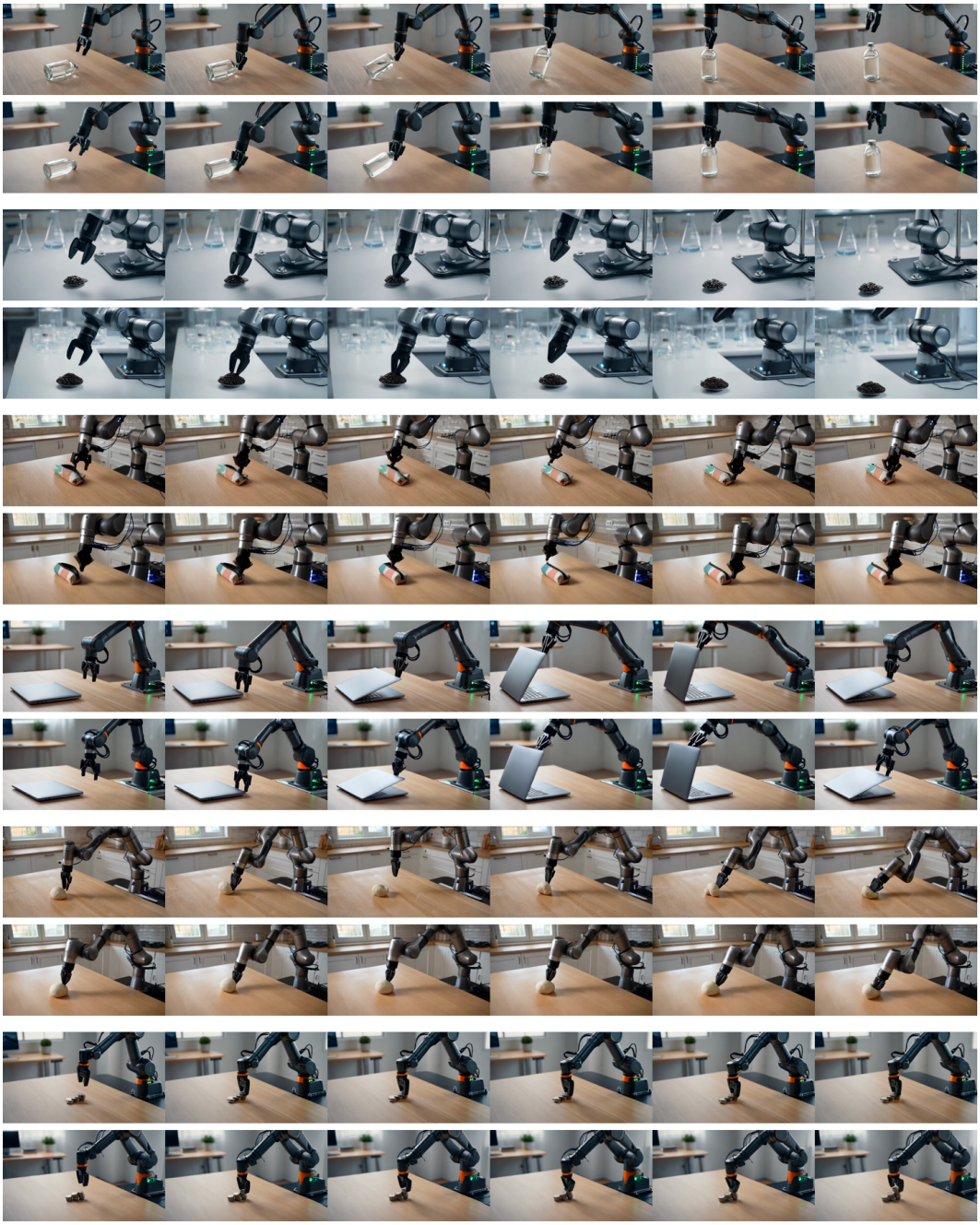


Figure 8: **Cosmos3-Super visual demo comparisons.** The video pairs compare the unaccelerated SGLang deployment with the Sol-Engine accelerated pipeline. The upper strip shows the unaccelerated SGLang result and the lower strip shows the Sol-Engine result. With $2.27\times$ end-to-end acceleration over SGLang, the accelerated pipeline preserves the main scene layout, object structure, and motion evolution.

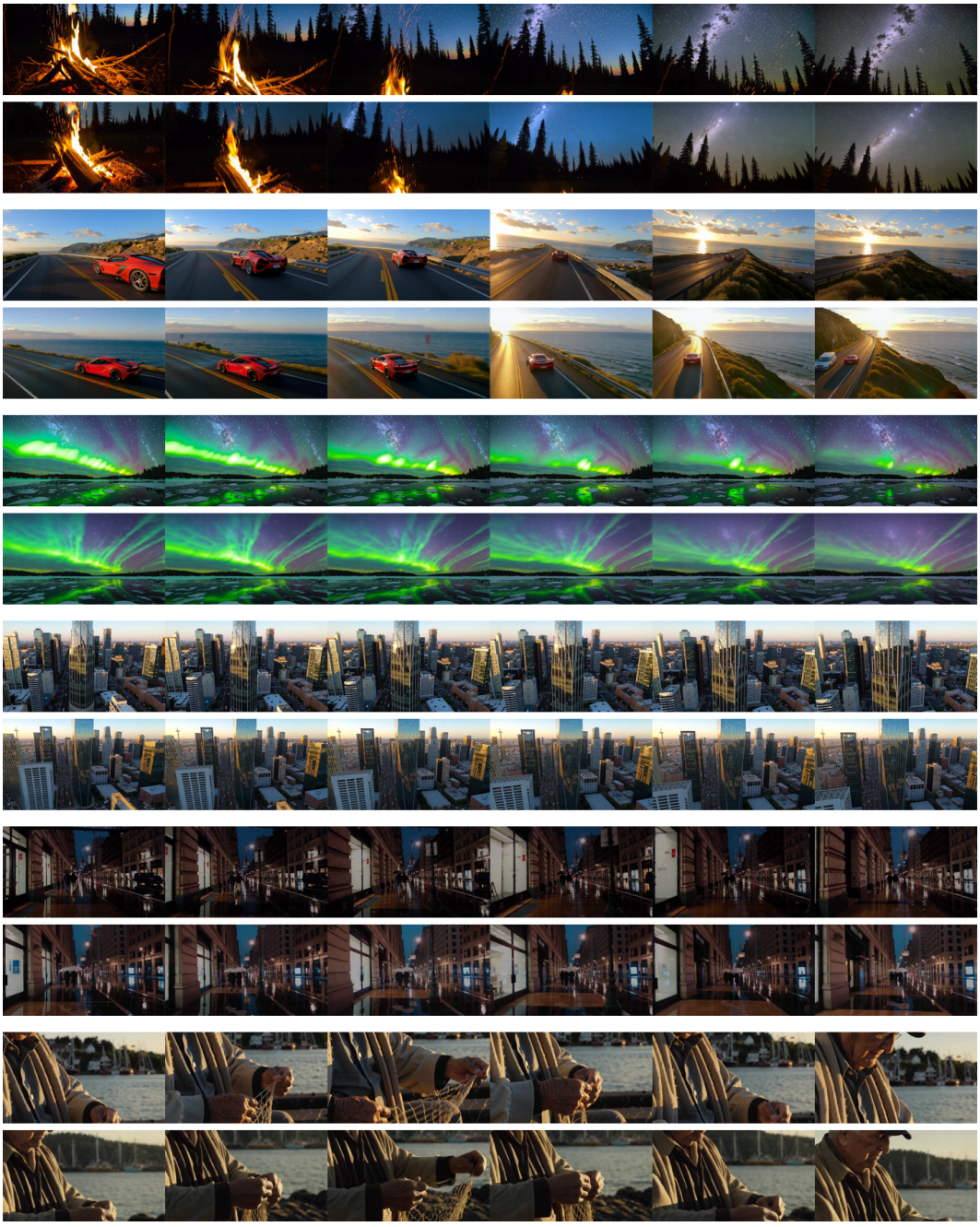


Figure 9: **LTX-2.3 visual demo comparisons.** The video pairs compare the unaccelerated SGLang deployment with the Sol-Engine accelerated pipeline. The upper strip shows the unaccelerated SGLang result and the lower strip shows the Sol-Engine result. With $2.38\times$ end-to-end acceleration over SGLang, the accelerated pipeline maintains the overall visual quality, temporal coherence, and scene composition.

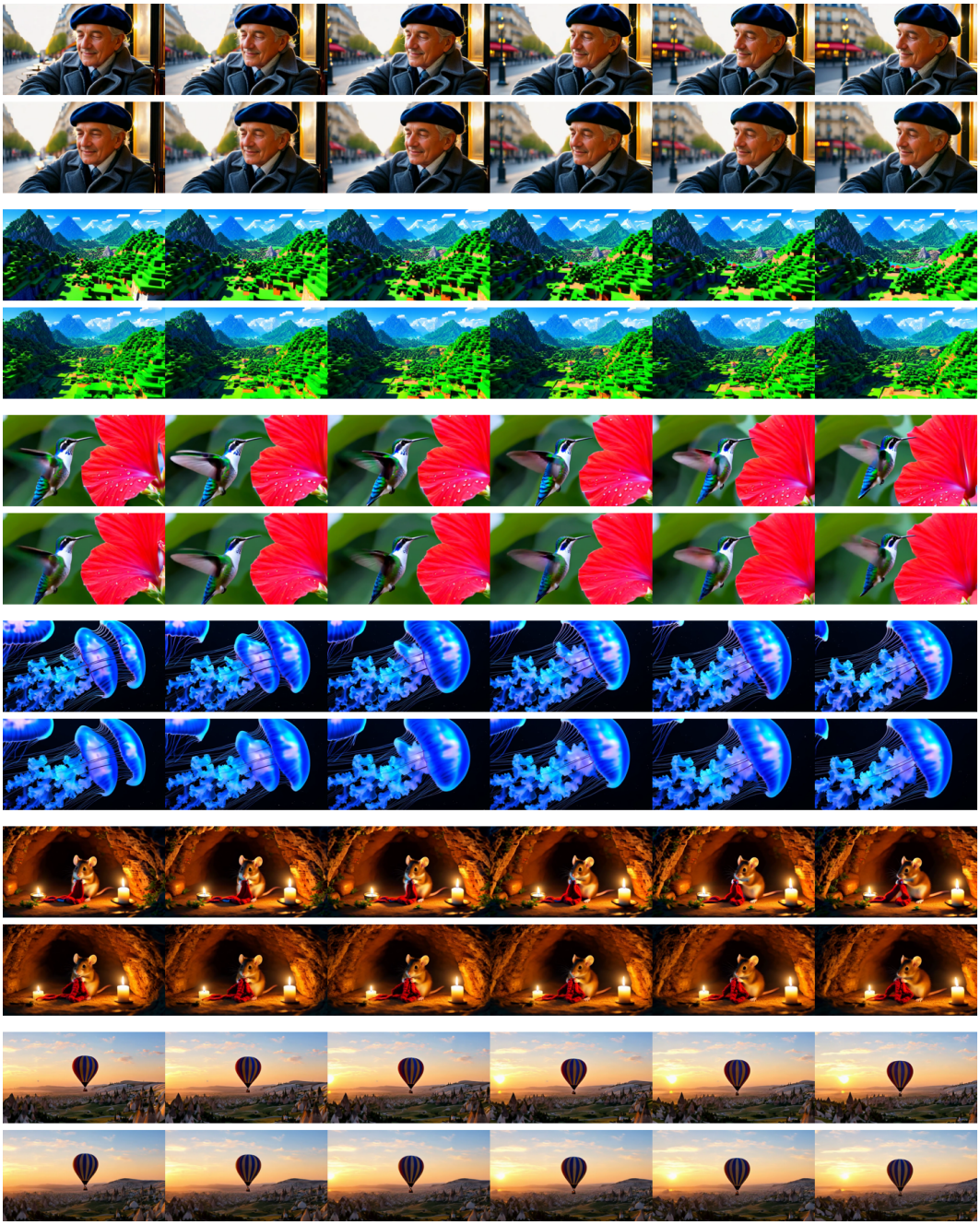


Figure 10: **SANA-Video visual demo comparisons.** The video pairs compare the unaccelerated SGLang deployment with the Sol-Engine accelerated pipeline. The upper strip shows the unaccelerated SGLang result and the lower strip shows the Sol-Engine result. With 2.77× end-to-end acceleration over SGLang, the accelerated pipeline preserves prompt-level appearance, motion trends, and perceptual quality.

F EXTENSION TO NVIDIA B300

We further evaluate single-GPU Cosmos3-Super on NVIDIA B200 and B300. Relative to B200, B300 maintains similar BF16 compute capability while providing higher NVFP4 compute capability (NVIDIA, 2026b). Accordingly, LTX-2.3 and SANA-Video, which rely less on NVFP4-heavy execution, show little end-to-end change in our single-GPU setting. By contrast, Cosmos3-Super depends more on NVFP4 execution and therefore obtains a clearer end-to-end speedup on B300. Table 3 reports single-GPU latency and speedup normalized to the baseline on the same card. On B200, Sol-Engine reduces latency from 353.5 s to 143.8 s (2.46×); on B300, it reduces latency from 351.9 s to 137.6 s (2.56×).

Table 3: **Single-GPU Cosmos3-Super on B200 and B300.** Speedups are normalized to the baseline on the same hardware. It shows a clearer B300 acceleration advantage with the NVFP4 stack.

Framework	Latency (s)	Speedup
B200 × SGLang	353.5	**1.00**×
B200 × Sol-Engine	143.8	**2.46**×
B300 × SGLang	351.9	**1.00**×
B300 × Sol-Engine	137.6	**2.56**×

G EXTENSION TO MINIMAX-H3 AND LTX-2.5

After the main experiments, we applied the same agentic workflow to two newly released audio-video models, MiniMax-H3 (MiniMax, 2026) and LTX-2.5 (Lightricks, 2026b), and deployed them on data-center, desktop, and edge GPUs. All results are training-free: the stock checkpoints are used without distillation, LoRA, fine-tuning, or offline calibration. Speedups are warm end-to-end latencies measured against an unoptimized baseline on the same hardware and GPU count, with the prompt, seed, resolution, number of frames, and number of steps held fixed; absolute latencies are not comparable across platforms.

G.1 MINIMAX-H3

Model and setting. MiniMax-H3 is a 33B dense omni-modal diffusion transformer that takes text, images, video, and audio as one context and generates video together with synchronized stereo audio. Its cost profile differs from the models in Sec. 4 in two ways. First, attention runs over a joint multimodal sequence of 38,247 tokens at 1344×768 and 124 frames, so attention dominates the denoiser. Second, roughly 13B of the 33B parameters lie in AdaLN modulation branches that run at every denoising step. Unless stated otherwise, we generate 124 frames (about 5 s at 24 fps) with 50 denoising steps and batch size 1.

Resulting stack. The agents converge on four components. (1) *Lossless kernel optimization*: fused RMS-AdaLN, which gives the largest single kernel-level gain because of the large AdaLN share; fused QK-norm with RoPE; merging the Q, K, and V projections into one GEMM; GEMM+GELU fusion; and `torch.compile` graph capture, including the VAE. (2) *Sparse attention* with Sol-Attn (Li et al., 2026a), a training-free sparse attention that selects key–value blocks during the online-softmax pass. (3) *Cross-step cache* with First Block Cache, which skips the remaining transformer blocks when the residual of the first block changes little between steps. (4) *Context parallelism* for multi-GPU deployments. The search for the 8-GPU GB200 deployment took about 4.5 hours. On H100 and A100, the model-level policy is shared: the first 10 steps and the first two layers stay dense, and Sol-Attn keeps an exact 951-token multimodal prefix; the resulting route density is 18.04% and the cache reuse rate is 67.35% on both platforms. Only the sparse-attention kernel is specialized to the hardware: a CuTe kernel on H100, a Triton kernel on A100, and a CuTe DSL kernel on RTX 4090.

Results across platforms. Tab. 4 reports the end-to-end speedup on six platforms, which ranges from 3.55× to 4.52×. Tab. 5 decomposes the speedup on three of them. The cross-step cache contributes the largest single step on every platform, whereas the gain from kernel optimization depends on the hardware: 1.39× on GB200 and DGX Spark but only 1.08× on RTX 5090. On GB200, the stack is offered in two configurations: *quality preserving* (cache without sparse attention, 3.16×, LPIPS 0.210 to the reference) and *super efficient* (with Sol-Attn, 3.95×, LPIPS 0.293). The

final configuration is also 2.80× faster than an SGLang deployment of the same model (19.30 s). Fig. 11 shows qualitative comparisons.

Table 4: **MiniMax-H3 across GPU platforms.** Warm end-to-end latency for 124 frames, 50 steps, and batch size 1. Each speedup is relative to the baseline on the same platform.

Platform	GPUs	Resolution	Baseline (s)	Ours (s)	Speedup
GB200	8	1344×768	27.21	6.88	3.95×
H100	4	1344×768	81.47	22.89	3.56×
A100	4	1344×768	217.32	61.28	3.55×
GeForce RTX 5090	1	1344×768	1045.4	231.2	4.52×
GeForce RTX 4090	1	1344×768	2239.22	504.33	4.44×
DGX Spark (GB10)	1	832×480	710.6	181.3	3.92×

Table 5: **Speedup breakdown on MiniMax-H3.** Cumulative end-to-end latency in seconds (speedup) as each technique is added. The GB200 baseline is the Diffusers reference implementation.

Stage	8×GB200, 768p	DGX Spark, 480p	RTX 5090, 768p
Baseline	27.21 (1.00×)	710.6 (1.00×)	1045.4 (1.00×)
+ Kernel optimization	19.51 (1.39×)	511.8 (1.39×)	971.6 (1.08×)
+ Sol-Attn	17.74 (1.53×)	451.2 (1.57×)	708.3 (1.48×)
+ Cross-step cache	6.88 (3.95×)	181.3 (3.92×)	231.2 (4.52×)

G.2 LTX-2.5

LTX-2.5 is a 22B diffusion transformer that generates video and synchronized audio. Its multi-step pipeline runs a 30-step guided first stage at low resolution, upsamples the latent, and finishes with a two-step distilled second stage at full resolution; its distilled pipeline uses 8 and 3 steps without classifier-free guidance. For the multi-step pipeline on 4×B200 in BF16, the agents replace the official sequence-parallel execution with parallelism over the four guidance branches, because the first-stage sequence is short enough that sequence parallelism is dominated by communication; they then add First Block Cache and kernel fusion with cache-aware compilation. For the NVFP4 distilled pipeline at 4K, attention in the high-resolution second stage dominates, and the agents apply Sol-Attn to that stage. Tab. 6 summarizes the results, and Fig. 12 shows qualitative comparisons.

Table 6: **LTX-2.5 results.** Warm end-to-end latency. Top: multi-step pipeline on 4×B200 (BF16), relative to the official sequence-parallel baseline. Bottom: NVFP4 distilled pipeline at 4K for 5 s on one GPU; the second-stage speedup is given in parentheses.

Pipeline	Setting	Baseline (s)	Ours (s)	Speedup
Multi-step	1536×1024, 5 s	39.23	8.38	4.68×
Multi-step	3840×2176, 5 s	75.57	38.49	1.96×
Multi-step	1920×1088, 20 s	92.45	36.69	2.52×
Distilled	B200	92.16	60.06	1.53× (1.80×)
Distilled	GeForce RTX 5090	322.81	170.32	1.90× (3.09×)
Distilled	DGX Spark (GB10)	829.60	573.80	1.45× (1.85×)

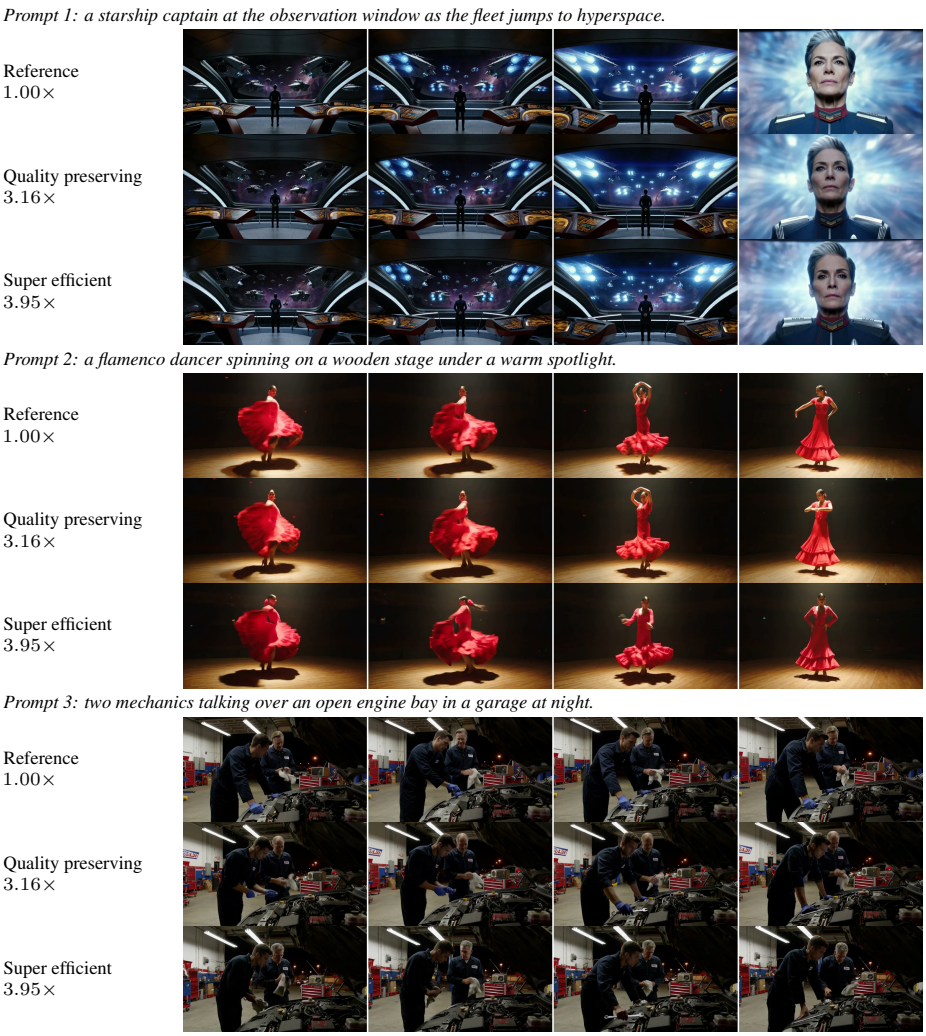


Figure 11: **MiniMax-H3 on 8×GB200.** Four frames sampled uniformly from each 1344×768, 124-frame clip, generated with the same prompt and seed. The generated audio is not shown.



Figure 12: **LTX-2.5 multi-step pipeline on 4×B200.** Two prompts at 1536×1024 for 5 s, generated with the official baseline and with our stack using the same prompt and seed.