Sparse Evolution: Efficient Test-Time Self-Evolution for Few-Step Video Generation

Anonymous authors
Paper under double-blind review

Abstract

Few-step distillation substantially reduces the cost of video generation, yet test-time search remains expensive due to repeated full-video candidate generation. Unlike final video synthesis, candidate selection primarily requires reliable relative rankings, suggesting that generating every candidate in full may be unnecessary. We find that, after a single full-sequence warm-up step, sparse keyframe previews retain discriminative information and largely preserve the rankings of their full-video counterparts. Building on this observation, we introduce Sparse Evolution, an efficient test-time self-evolution framework for few-step video generation. Sparse Evolution adaptively selects keyframes using warm-up features and restricts subsequent denoising and decoding to the retained temporal positions, reducing computation. The resulting sparse previews guide candidate selection and provide evidence for iterative prompt refinement, with full-video generation completed only for the selected prompt–seed pair. Across three few-step video generators, sparse previews achieve mean per-prompt Spearman correlations of 0.75–0.81 with full candidates, while Sparse Evolution accelerates candidate generation by 2.00–2.25× over SVG2. On T2V-CompBench and VBench Semantic, Sparse Evolution achieves 1.74–1.91× end-to-end speedup while retaining 91% of the quality gains from test-time search.

1 Introduction

As demand for low-latency video generation grows, few-step distillation compresses tens of denoising steps into just a few, substantially reducing per-video latency (Yin et al., 2025; Fan et al., 2025). Yet fewer state updates leave less opportunity to satisfy instruction constraints, so complex instructions involving multi-subject interactions, attribute binding, and action completion remain challenging (Sun et al., 2025). To improve instruction following without retraining, test-time scaling (TTS) compares multiple candidates and may refine prompts using feedback (Liu et al., 2025; Long et al., 2026; Song et al., 2026). However, generating every candidate in full imposes all denoising and decoding costs even on discarded samples, causing search cost to grow with the candidate count and number of refinement rounds.

A further way to reduce search cost is to lower the generation cost of each candidate during exploration. CachedSearch (Saini et al., 2026) reuses intermediate features across denoising steps and regenerates only the selected seed using full computation. However, its gains depend on cross-step redundancy, which is limited in few-step models. Moreover, each remaining step still processes temporal positions spanning the entire video. We therefore investigate temporal sparsification to reduce candidate exploration cost while preserving sufficient information for reliable comparison (Figure 1).

For temporal sparsification to support reliable candidate selection, sparse candidates must preserve the relative ranking of full videos. Using prompt- and seed-matched sparse–full pairs with a shared full-sequence warm-up step, we find mean per-prompt Spearman correlations of 0.75–0.81 across three few-step models, as shown in Figure 2. As more temporal positions are retained, ranking correlation saturates while generation cost continues to rise; we therefore select operating points that retain only 43%–48% of latent temporal positions.

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.

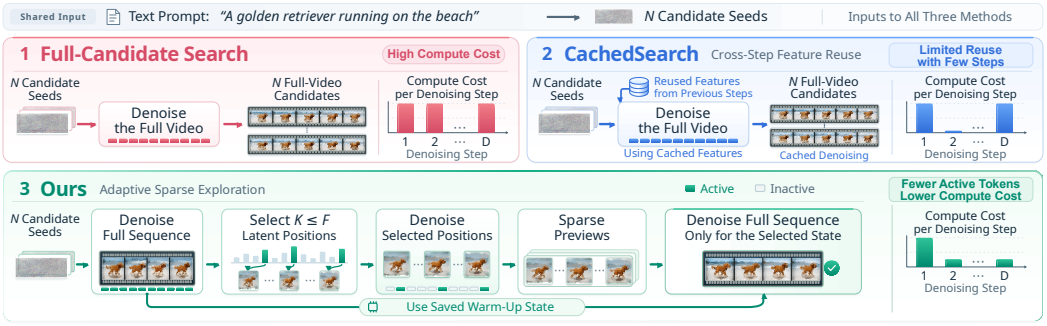

Figure 1: Search acceleration beyond denoising-step reuse. Full-video search and CachedSearch process complete temporal sequences, whereas Sparse Evolution reduces temporal computation during candidate exploration. After one full-sequence warm-up step, it generates sparse previews and completes only the selected candidate. The right-hand plots schematically illustrate per-step candidate-generation costs.

Based on this observation, we propose **Sparse Evolution**, an efficient test-time self-evolution framework for few-step video generation. The method adaptively selects temporal positions from the warm-up prediction and restricts subsequent denoising and decoding to the retained positions, producing low-cost previews for both best-of-N search and iterative prompt refinement. Once search terminates, only the selected candidate is completed into a full video by resuming from its saved full-sequence warm-up state.

On T2V-CompBench and VBench Semantic, Sparse Evolution achieves 1.75–1.92× generation speedups over full-candidate Best-of-8 while retaining, on average, 91.1% and 91.8% of its search-quality gains, respectively. Compared with CachedSearch, it provides 1.60–1.74× speedups with similar final-video quality. The same sparse candidate backend also accelerates iterative prompt refinement and integrates into VQQA.

Our contributions are threefold:

- **Sparse search proxy analysis.** We systematically study whether temporally sparse candidates can preserve the rankings of full-video candidates and characterize the trade-off among temporal budget, ranking consistency, and computation.
- **Adaptive sparse candidate generation.** We introduce a candidate-generation mechanism that adaptively selects informative temporal positions and restricts subsequent denoising and decoding to the retained subset, substantially reducing the spatiotemporal computation required for candidate evaluation.
- **Efficient test-time self-evolution for few-step video generation.** We apply sparse previews to both best-of-N search and iterative prompt refinement, demonstrating favorable quality–efficiency trade-offs across three few-step video generation configurations and two benchmarks.

2 Method

2.1 Preliminaries

Few-step video generation. Given a text prompt p and a random seed s, a latent video generator starts from an initial noise sample $z^{(0)}(s)$, performs D denoising steps, and decodes the resulting latent with a video VAE:

$$y^F = G_F(p, s; D) = \mathrm{Dec}\left(z^{(D)}\right). \tag{1}$$

Here, $z^{(d)} = (z_1^{(d)}, \ldots, z_F^{(d)})$ denotes the latent state after step d, and F is the latent temporal length. All temporal sparsification in this work operates on these latent positions.

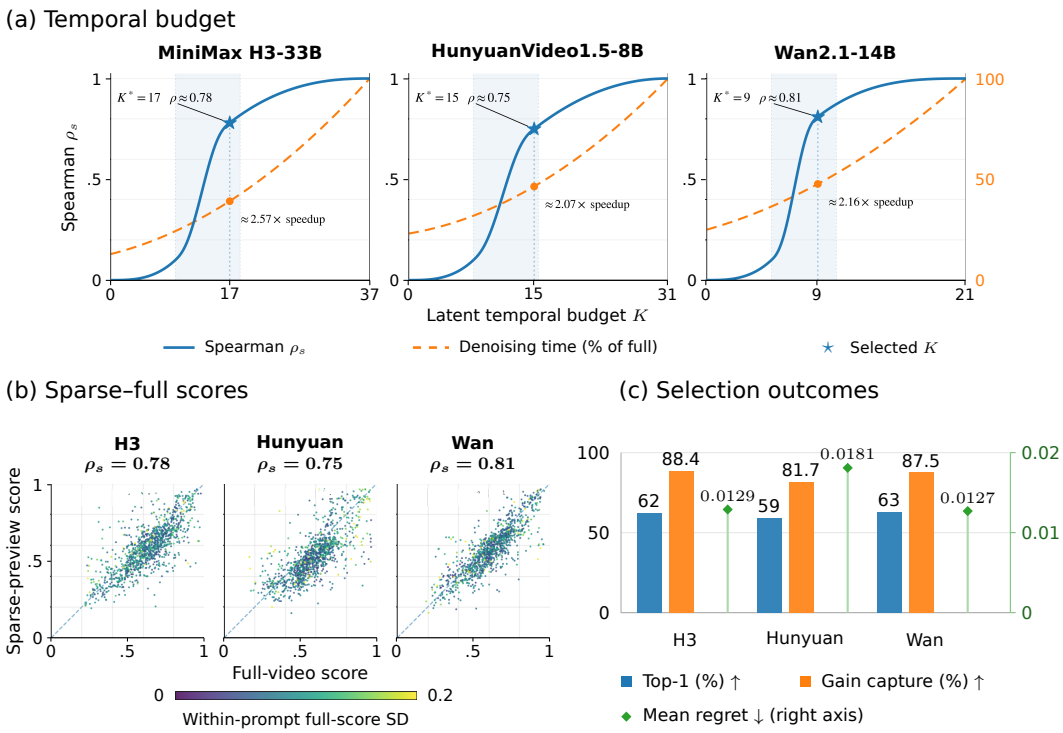

Figure 2: **Effective candidate selection can rely on sparse temporal evidence.** (a) Ranking correlation gradually saturates while denoising cost continues to grow; stars mark the selected budgets. (b) Paired sparse–full scores exhibit strong within-prompt ranking agreement. Each point represents one candidate; color denotes the within-prompt standard deviation of full-video scores, and ρ_s is averaged over prompts. (c) Sparse selection retains most full-search gains despite imperfect Top-1 agreement. Top-1 and gain capture use the left axis; mean Regret uses the right. Results use 140 prompts and eight seeds per prompt for each model.

Test-time search. Given an original prompt p_0, test-time search explores a set of prompt–seed pairs $\{(p_i, s_i)\}_{i=1}^{N}$. Conventional full-video search generates each candidate and evaluates its alignment with the original prompt using a scoring function V:

$$q_i = V(G_F(p_i, s_i; D), p_0), \qquad i^\star = \arg\max_i q_i. \tag{2}$$

In best-of-N search, $p_i = p_0$, and candidates differ only in their seeds; in iterative prompt refinement, p_i is updated using generation feedback. Both settings evaluate candidates against p_0. The experimental configuration of V is detailed in Section 3.1.

2.2 Sparse Video as a Search Proxy

Temporal sparsification changes both the generation trajectory and the content available for evaluation, so a visually plausible preview may not reflect the quality of its full-video counterpart. To assess its ranking fidelity, we construct prompt- and seed-matched sparse–full candidate pairs that share one full-sequence warm-up step. The full branch then processes all F latent temporal positions, whereas the sparse branch continues denoising only K adaptively selected positions.

Temporal budget and ranking consistency. We vary K while keeping the candidate set and evaluation protocol fixed, using mean per-prompt Spearman correlation to measure ranking consistency and examine its trade-off with denoising cost. As shown in Figure 2(a), increasing K improves ranking correlation with diminishing returns, while denoising cost continues to grow. We therefore select $K = 17$, 15, and 9 for MiniMax H3-33B, HunyuanVideo1.5-8B, and Wan2.1-14B, respectively, retaining 43%–48% of the latent temporal positions during subsequent denoising.

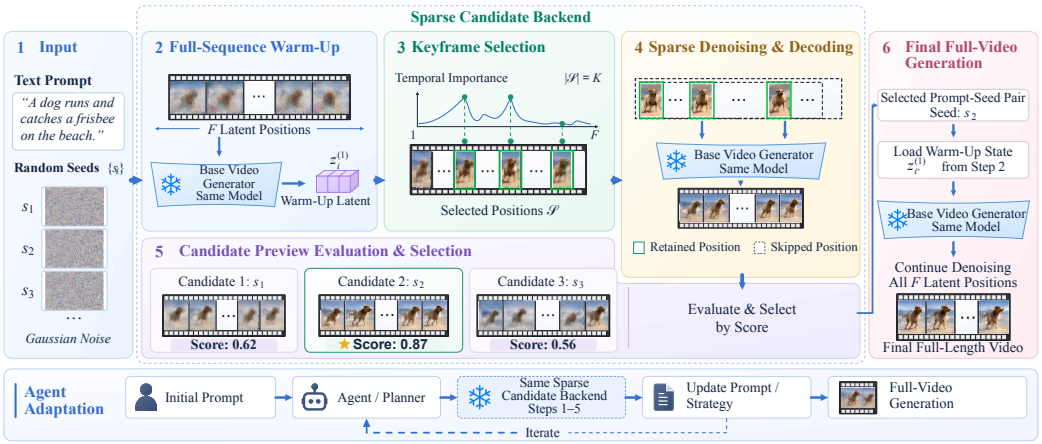

Figure 3: **Overview of Sparse Evolution: decoupling candidate exploration from final video synthesis.** After one full-sequence warm-up step, subsequent denoising and decoding are restricted to adaptively selected temporal positions, producing low-cost previews for best-of-N search. Only the selected prompt–seed pair is completed into a full video by resuming from its saved full-sequence warm-up state. The bottom panel shows how the same candidate backend supports iterative prompt refinement.

From ranking preservation to selection quality. At these budgets, the three models achieve mean per-prompt Spearman correlations of 0.78, 0.75, and 0.81, respectively (Figure 2(b)). To connect ranking consistency to search decisions, we next examine whether sparse evaluation selects a candidate with the highest full-video verifier score. Top-1 agreement measures the fraction of prompts for which sparse evaluation selects any seed attaining the highest full-video score, including ties. The corresponding agreement rates are 62%, 59%, and 63% (Figure 2(c)).

Top-1 agreement alone does not reveal the quality loss when a different candidate is selected. Mean Regret measures the cost of selection differences: for each prompt, we subtract the full-video score of the sparse-selected seed from the maximum full-candidate score, then average across prompts. The three models obtain Regret values of 0.0129, 0.0181, and 0.0127 on a $[0, 1]$ score scale, retaining 88.4%, 81.7%, and 87.5% of the full-search gains, respectively (Figure 2(c)). Despite imperfect selection agreement, sparse previews retain most of the gains from full-candidate search.

2.3 Sparse Evolution Framework

These findings motivate a separation between the computation needed to compare candidates and that required to produce the final video. Sparse Evolution uses low-cost keyframe previews for best-of-N search and iterative prompt refinement, completing full-video generation only for the selected candidate (Figure 3).

2.3.1 Sparse Candidate Generation

Full-sequence warm-up. For each prompt–seed candidate, we execute the first denoising step over all F latent temporal positions using the base generator's noise initialization, text condition, and sampling configuration, and save the resulting state $z_i^{(1)}$ together with the sampler state. The same forward pass provides a predicted clean latent $\hat{z}_{0,i}$ for content-adaptive frame selection across the full sequence.

Adaptive frame selection. Given a temporal budget $1 \le K \le F$, we use the first seed in a predefined candidate ordering as the reference. We spatially downsample and flatten each latent temporal position of its warm-up prediction to obtain a descriptor h_f. Content

differences between positions are measured by cosine distance:

$$d(f, a) = 1 - \frac{h_f^\top h_a}{\|h_f\|_2 \|h_a\|_2}. \tag{3}$$

Starting from $\mathcal{S} = \{1\}$, we iteratively add the unretained position with the largest minimum distance to the retained set:

$$f^\star = \arg\max_{f \notin \mathcal{S}} \min_{a \in \mathcal{S}} d(f, a), \qquad \mathcal{S} \leftarrow \mathcal{S} \cup \{f^\star\}. \tag{4}$$

We stop once $|\mathcal{S}| = K$, break ties in favor of earlier positions, and sort the selected indices in temporal order. Candidates for a prompt share $\mathcal{S}$ but retain separate warm-up states.

Sparse denoising and preview generation. For each candidate, we extract $z^{(1)}_{i,\mathcal{S}}$ from its own full warm-up state, pack the selected positions in chronological order into a tensor of shape $[B, C, K, H, W]$, and perform the remaining $D-1$ denoising steps. Retained video tokens supply queries, keys, and values and undergo the base model's attention, feed-forward, and residual updates. Unselected positions are excluded from subsequent computation, reducing attention, projection, and feed-forward costs. Temporal RoPE indices are re-indexed as $0, \ldots, K-1$ in packed order, while spatial indices remain unchanged. The denoised sparse latent is jointly decoded by the native temporal VAE into a shorter RGB preview for verifier scoring, without reconstructing the full timeline or inserting missing frames.

2.3.2 Preview-Based Search and Full-Video Completion

Best-of-N search. For a fixed original prompt p_0, we generate sparse previews for N seeds, score them using $V(\cdot, p_0)$, and select the highest-scoring candidate.

Full-video completion. We restore the selected candidate's saved full-sequence latent $z^{(1)}$ and sampler state, perform the remaining $D-1$ denoising steps over all F positions, and decode the result. Generation resumes from the full warm-up state rather than the sparse preview, avoiding reconstruction of omitted positions and repetition of the first step.

Iterative prompt refinement. Sparse previews can replace full candidates in agent search while preserving the existing prompt-update, exploration, and stopping rules; all scores remain conditioned on the original prompt p_0. Once search terminates, the selected prompt–seed pair is completed using the procedure above. Appendix D analyzes costs.

3 Experiments

Building on the verifier-based ranking analysis in Section 2.2, we now examine whether sparse candidate selection preserves gains in final-video benchmark scores. We evaluate search quality and efficiency through best-of-N search and iterative prompt refinement, followed by ablations of the key design choices.

3.1 Experimental Setup

Models and settings. We evaluate MiniMax H3-33B (H3) (MiniMax, 2026), HunyuanVideo1.5-8B (Hunyuan) (Tencent Hunyuan Foundation Model Team, 2025), and Wan2.1-14B (Wan) (Team Wan et al., 2025) with 8, 8, and 4 sampling steps, respectively, on NVIDIA H100 GPUs using BF16 DiT backbones. For each model, methods share weights, samplers, resolutions, output lengths, model residency, and request concurrency.

Benchmarks and metrics. Final full-length videos are evaluated against the original prompts using the seven-category mean of T2V-CompBench (Sun et al., 2025) and the official VBench Semantic aggregation (Huang et al., 2024), with scores scaled by 100. Search-quality gains are measured relative to the same model's few-step single-sample baseline.

Verifier protocol. Section 2.2 and Table 4 use 140 prompts sampled from T2V-CompBench, with eight seeds per prompt. These prompts are used for ranking analysis



Figure 4: Final-video comparison of Few-Step Baseline, Best-of-8, CachedSearch, and Sparse Evolution under fixed prompts. Few-Step Baseline: single-sample generation without search.

and selecting K, and are excluded from the final evaluation in Tables 1–3. GPT-5.5 generates VQA questions from the original prompt p_0 and evaluates sparse previews and full videos under the same protocol, with frames sampled following the T2V-CompBench preprocessing rule. Ranking metrics and the $[0, 1]$ verifier score scale follow Section 2.2. Verifier and agent configurations are provided in Appendices A and H.

Timing protocol. Tables 1–3 report directly measured per-prompt generation latency with generator weights resident, covering candidate exploration and final-video denoising and decoding. Table 4 reports directly measured generation latency per candidate, including warm-up, frame selection, and decoding. Request-level wall-clock timing, including verifier and agent calls, is described in Appendix C.

3.2 Best-of-N Search Results

To isolate the effect of candidate generation, we disable iterative prompt refinement and compare search with full, cached, and keyframe candidates, using 50-step and few-step single-sample generation as quality references.

Sparse Evolution substantially reduces candidate exploration cost while preserving most of the quality gains from full-candidate search. Compared with Best-of-8, it achieves a 1.75–1.92× generation speedup while retaining, on average, 91.1% and 91.8% of its search-quality gains on T2V-CompBench and VBench Semantic, respectively (Table 1).

The Best-of-8 comparison fixes the candidate count; Best-of-4 tests the alternative of reducing the number of full candidates. Compared with Best-of-4, Sparse Evolution explores twice as many candidates at 4.3%–14.5% higher generation cost, improving scores on the two benchmarks by 1.2–2.4 and 0.6–1.0 points, respectively. This highlights the value of low-cost previews for broader candidate exploration at comparable computational cost.

Sparse Evolution is particularly effective for few-step search, where the opportunities for cross-step reuse are limited. Compared with CachedSearch (Saini et al., 2026), it achieves a 1.60–1.74× speedup while maintaining similar final quality. Figure 4 presents qualitative comparisons. CachedSearch relies on redundancy across denoising steps, whereas Sparse Evolution reduces the temporal tokens processed in subsequent steps, allowing further acceleration even with 4- and 8-step generators.

3.3 Iterative Prompt Refinement and Cross-Framework Adaptation

Unlike best-of-N search, iterative prompt refinement uses candidate evidence not only for selection but also to guide subsequent prompts. We keep evaluation, prompt-update, and stopping rules fixed and vary only the candidate backend, comparing Full-Agent, Cached-Agent, and Sparse-Agent. We further integrate the sparse candidate backend into VQQA (Song et al., 2026) to assess its applicability across frameworks.

Table 1: Fixed-prompt results on T2V-CompBench and VBench Semantic. Latency is in s/prompt; speedup is relative to Best-of-8.

Model	Method	T2V-CompBench ↑	VBench Semantic ↑	Latency ↓	Speedup ↑
H3	50-step	53.5	76.4	483.1	—
	8-step	51.8	75.3	78.5	—
	Best-of-8	57.4	77.9	628.0	1.00×
	Best-of-4	55.5	77.0	314.0	2.00×
	CachedSearch	56.9	77.7	570.9	1.10×
	Sparse Evolution	56.7	77.6	327.4	1.92×
Hunyuan	50-step	47.7	62.1	430.9	—
	8-step	45.3	61.7	47.4	—
	Best-of-8	53.0	64.9	379.2	1.00×
	Best-of-4	50.1	63.8	189.6	2.00×
	CachedSearch	52.6	64.7	354.4	1.07×
	Sparse Evolution	52.5	64.7	217.1	1.75×
Wan	50-step	50.6	75.2	194.7	—
	4-step	49.9	74.6	21.9	—
	Best-of-8	55.1	77.5	175.2	1.00×
	Best-of-4	52.8	76.3	87.6	2.00×
	CachedSearch	54.5	77.2	157.8	1.11×
	Sparse Evolution	54.7	77.3	98.9	1.77×

Table 2: Candidate backend comparison under the same agent search strategy. Latency is in s/prompt; speedup is relative to Full-Agent.

		T2V-CompBench			**VBench Semantic**		
Model	Method	Quality ↑	Latency ↓	Speedup ↑	Quality ↑	Latency ↓	Speedup ↑
MiniMax H3-33B	Full-Agent	60.3	543.3	1.00×	80.8	413.1	1.00×
	Cached-Agent	59.8	493.9	1.10×	80.5	379.0	1.09×
	Sparse-Agent	59.5	292.1	1.86×	80.3	238.8	1.73×
HunyuanVideo 1.5-8B	Full-Agent	57.1	328.9	1.00×	67.4	247.6	1.00×
	Cached-Agent	57.1	307.4	1.07×	67.2	233.6	1.06×
	Sparse-Agent	57.2	193.5	1.70×	67.1	156.7	1.58×
Wan2.1 14B	Full-Agent	59.1	151.5	1.00×	79.9	115.2	1.00×
	Cached-Agent	58.5	136.5	1.11×	79.1	102.8	1.12×
	Sparse-Agent	58.6	87.6	1.73×	79.2	70.6	1.63×

3.3.1 Gains within Our Agent Framework

Compared with Full-Agent, Sparse-Agent achieves generation speedups of 1.70–1.86× and 1.58–1.73× on T2V-CompBench and VBench Semantic, respectively, while retaining, on average, 95.3% and 90.8% of Full-Agent's search-quality gains. Compared with Cached-Agent, the corresponding speedups are 1.56–1.69× and 1.46–1.59×, with similar final quality (Table 2). Measured end-to-end results, including verifier and agent calls, are reported in Appendix C.1. Sparse previews also preserve most of the gains from evidence-guided refinement (Appendix G).

3.3.2 Integrating Sparse Previews into VQQA

Keeping the few-step generator and VQQA search procedure unchanged, replacing full candidates with sparse previews yields generation speedups of 1.39–1.50× and 1.34–1.46× on T2V-CompBench and VBench Semantic, respectively, while retaining 88.4% of the original search-quality gains on average (Table 3). End-to-end results are reported in Appendix C.1.

Table 3: Results of integrating sparse previews into VQQA. Latency is in s/prompt.

		T2V-CompBench			VBench Semantic		
Model	Method	Quality ↑	Latency ↓	Speedup ↑	Quality ↑	Latency ↓	Speedup ↑
MiniMax H3-33B	VQQA	55.3	212.0	1.00×	78.2	196.3	1.00×
	VQQA + Sparse	54.7	152.5	1.39×	77.9	146.5	1.34×
HunyuanVideo 1.5-8B	VQQA	53.8	156.4	1.00×	65.4	146.9	1.00×
	VQQA + Sparse	53.6	104.3	1.50×	65.3	100.6	1.46×
Wan2.1 14B	VQQA	54.0	61.3	1.00×	76.9	56.9	1.00×
	VQQA + Sparse	53.2	41.2	1.49×	76.5	39.5	1.44×

Table 4: Ablation of candidate-generation backends and temporal selection strategies. Latency is in s/candidate; Top-1 and gain capture are percentages.

Model	Method	Score ↑	ρ_s ↑	Top-1 ↑	Regret ↓	Gain capture ↑	Latency ↓	Speedup ↑
H3	Dense	71.15	1.00	100	0.0000	100.0	78.50	1.00×
	SVG2	70.04	0.82	65	0.0111	90.1	72.59	1.08×
	Uniform	68.94	0.76	61	0.0221	80.1	37.01	2.12×
	Adaptive	69.86	0.78	62	0.0129	88.4	32.21	2.44×
Hunyuan	Dense	69.91	1.00	100	0.0000	100.0	47.40	1.00×
	SVG2	68.46	0.81	67	0.0145	85.4	43.64	1.09×
	Uniform	67.74	0.74	57	0.0217	78.0	22.99	2.06×
	Adaptive	68.10	0.75	59	0.0181	81.7	21.82	2.17×
Wan	Dense	70.21	1.00	100	0.0000	100.0	21.90	1.00×
	SVG2	69.26	0.86	72	0.0095	90.6	21.41	1.02×
	Uniform	68.30	0.78	61	0.0191	81.3	11.42	1.92×
	Adaptive	68.94	0.81	63	0.0127	87.5	10.21	2.14×

These results show that the sparse candidate backend reduces search cost beyond our own agent framework while preserving most of the gains from iterative prompt refinement.

3.4 Ablation Studies

Having evaluated the complete search pipeline, we next examine two design choices underlying its quality–cost trade-off: reducing the active temporal positions and selecting them according to content.

Ablation on candidate sparsification. We replace the sparse candidate backend with full-sequence sparse attention from SVG2 (Yang et al., 2025), keeping the candidate set, evaluation protocol, and search procedure fixed. Compared with SVG2, Sparse Evolution achieves a 2.00–2.25× candidate-generation speedup and ranking correlations of 0.75–0.81, with full-video verifier scores of the selected seeds only 0.18–0.36 points lower (Table 4). This shows that directly reducing temporal tokens further lowers generation cost with a small loss in selection quality.

Ablation on adaptive frame selection. Uniform uses the same sparse candidate backend and retains every other temporal position starting from the first, giving 19, 16, and 11 positions for H3, Hunyuan, and Wan, respectively. Adaptive retains only 17, 15, and 9 positions, yet achieves higher verifier scores while reducing candidate-generation latency by 13.0%, 5.1%, and 10.6%, respectively (Table 4). These results suggest that, rather than spreading a limited frame budget uniformly across the timeline, Adaptive concentrates computation on more discriminative temporal regions, yielding higher verifier scores with fewer retained positions. Comparisons at the same K and additional component ablations are provided in Appendices E and F.

4 Related Work

Efficient video generation. Existing approaches accelerate video generation primarily by reducing the number of sampling steps or reusing computation across denoising steps. CausVid (Yin et al., 2025) distills bidirectional video diffusion models into few-step autoregressive generators, while Phased DMD (Fan et al., 2025) improves few-step generation through phased distribution matching and score matching within subintervals. These methods reduce per-video sampling cost, which is repeatedly incurred across candidates in test-time search. Sparse Evolution targets this setting by further reducing temporal computation during candidate generation on top of few-step generators, without modifying the underlying distillation procedure.

Sparse computation for video generation. Training-free sparse methods mainly reduce the per-step cost of video diffusion transformers (DiTs) (Peebles & Xie, 2023) by sparsifying attention interactions among video tokens. SVG (Xi et al., 2025) exploits structured spatial and temporal sparsity patterns across attention heads, STA (Zhang et al., 2025) organizes local attention into hardware-friendly tiled computation, and SVG2 (Yang et al., 2025) further uses semantic-aware token permutation to form more compact sparse attention layouts. These methods primarily sparsify interactions between tokens while retaining the full temporal token sequence, so temporal positions that receive fewer attention connections still incur projection and feed-forward computation. Sparse Evolution instead produces sparse search previews: it excludes unselected temporal positions from subsequent denoising, so omitted tokens no longer participate in the remaining DiT computation.

Test-time search for video generation. Test-time methods improve generation quality by exploring and evaluating multiple candidates. Video-T1 (Liu et al., 2025) uses verifier-guided candidate expansion and pruning, while VISTA (Long et al., 2026), VQQA (Song et al., 2026), and GenMAC (Huang et al., 2026) perform iterative prompt refinement or update generation conditions using video evaluation, visual question answering, or multi-agent feedback. While effective, these methods repeatedly generate candidates, causing exploration cost to grow with the number of candidates and refinement rounds. Cached-Search (Saini et al., 2026) reduces this overhead by reusing features across denoising steps to construct approximate candidates and performs full computation only for the selected seed. However, its savings mainly come from redundancy across denoising steps, whereas few-step generators contain only a small number of such steps and each candidate still spans the full temporal sequence. Sparse Evolution instead constructs low-cost search proxies through temporal sparsification, making its candidate-generation savings independent of having many denoising steps available for cross-step reuse.

5 Conclusion

We introduced Sparse Evolution, a framework that improves test-time search efficiency for few-step video generation by separating low-cost candidate exploration from final full-video synthesis. After one full-sequence warm-up step, the method adaptively selects keyframes and restricts subsequent denoising and decoding to the retained temporal positions. Sparse previews support candidate scoring, selection, and search feedback, while the selected prompt–seed pair is completed by resuming from its saved full-sequence warm-up state, avoiding repeated warm-up computation.

Experiments show that sparse previews largely preserve full-candidate rankings and retain most search-quality gains at substantially lower generation cost. Compared with sparse attention over full temporal sequences, directly reducing the active temporal positions further improves the quality–efficiency trade-off. The same candidate-generation backend also supports iterative prompt refinement, demonstrating its applicability across search procedures. These results highlight the value of distinguishing the computational requirements of candidate evaluation from those of final video synthesis.

AI USE STATEMENT

We used ChatGPT to assist with translation and literature search. Separately, GPT-5.5 serves as the Planner, Verifier, and Analyzer in our experimental pipeline, as documented in Section 3.1 and Appendices A and H. The authors are responsible for verifying all AI-assisted material and take full responsibility for the final text, methods, experimental results, references, and conclusions.

REFERENCES

Xiangyu Fan, Zesong Qiu, Zhuguanyu Wu, Fanzhou Wang, Zhiqian Lin, Tianxiang Ren, Dahua Lin, Ruihao Gong, and Lei Yang. Phased DMD: Few-step distribution matching distillation via score matching within subintervals. *arXiv preprint arXiv:2510.27684*, 2025. URL https://arxiv.org/abs/2510.27684.

Kaiyi Huang, Yukun Huang, Xuefei Ning, Zinan Lin, Yu Wang, and Xihui Liu. GENMAC: Compositional text-to-video generation with multi-agent collaboration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(7):5049–5057, 2026. doi: 10.1609/aaai.v40i7.37418. URL https://ojs.aaai.org/index.php/AAAI/article/view/37418.

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.

Fangfu Liu, Hanyang Wang, Yimo Cai, Kaiyan Zhang, Xiaohang Zhan, and Yueqi Duan. Video-T1: Test-time scaling for video generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18671–18681, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Liu_Video-T1_Test-time_Scaling_for_Video_Generation_ICCV_2025_paper.html.

Do Xuan Long, Xingchen Wan, Hootan Nakhost, Chen-Yu Lee, Tomas Pfister, and Sercan Ö. Arik. VISTA: A test-time self-improving video generation agent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6021–6032, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Long_VISTA_A_Test-Time_Self-Improving_Video_Generation_Agent_CVPR_2026_paper.html.

MiniMax. MiniMax-H3. Official model card, 2026. URL https://huggingface.co/MiniMaxAI/MiniMax-H3.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Peebles_Scalable_Diffusion_Models_with_Transformers_ICCV_2023_paper.html.

Shreshth Saini, Neil Birkbeck, Yilin Wang, Balu Adsumilli, and Alan C. Bovik. CachedSearch: Training-free cached exploration for test-time search in video diffusion. *arXiv preprint arXiv:2607.23159*, 2026. URL https://arxiv.org/abs/2607.23159.

Yiwen Song, Tomas Pfister, and Yale Song. VQQA: An agentic approach for video evaluation and quality improvement. *arXiv preprint arXiv:2603.12310*, 2026. URL https://arxiv.org/abs/2603.12310.

Kaiyue Sun, Kaiyi Huang, Xian Liu, Yue Wu, Zihan Xu, Zhenguo Li, and Xihui Liu. T2V-CompBench: A comprehensive benchmark for compositional text-to-video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8406–8416, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Sun_T2V-CompBench_A_Comprehensive_Benchmark_for_Compositional_Text-to-video_Generation_CVPR_2025_paper.html.

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. *arXiv preprint arXiv:2503.20314*, 2025. URL https://arxiv.org/abs/2503.20314.

Tencent Hunyuan Foundation Model Team. HunyuanVideo 1.5 technical report. *arXiv preprint arXiv:2511.18870*, 2025. URL https://arxiv.org/abs/2511.18870.

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 Video-Gen: Accelerating video diffusion transformers with spatial-temporal sparsity. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 68208–68224. PMLR, 2025. URL https://proceedings.mlr.press/v267/xi25c.html.

Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Chenfeng Xu, Kelly Peng, Jianfei Chen, Song Han, Kurt Keutzer, and Ion Stoica. Sparse VideoGen2: Accelerate video generation with sparse attention via semantic-aware permutation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 107568–107594. Curran Associates, Inc., 2025. doi: 10.52202/085713-3243. URL https://papers.nips.cc/paper_files/paper/2025/hash/8bd5148caced2d73cea7b6961a874a49-Abstract-Conference.html.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T. Freeman, Fredo 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)*, pp. 22963–22974, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Yin_From_Slow_Bidirectional_to_Fast_Autoregressive_Video_Diffusion_Models_CVPR_2025_paper.html.

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. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhang25m.html.

A Model and Inference Configurations

Table A.1: Few-step generation configurations. All models use CFG 1.0 and one full-sequence warm-up step. Resolution is width × height. Output length is given in RGB frames; K counts retained latent temporal positions.

Model	Variant	Sampler	Steps	Resolution	Frames	K
MiniMax H3-33B	Turbo	FlowMatch	8	1344×768	124	17
HunyuanVideo1.5-8B	LightX2V	Euler	8	832×480	121	15
Wan2.1-14B	LightX2V	DMD2 Euler	4	832×480	81	9

Inference procedure. Algorithm A.1 summarizes the candidate backend for both best-of-N search and iterative prompt refinement. The temporal index set is shared across candidates, while each candidate retains its own full-sequence warm-up and sampler states for final completion.

Algorithm A.1 Sparse Evolution search and full-video completion

Require: Original prompt p_0; temporal budget K; D-step generator; verifier V; search policy Π
Ensure: Selected full video $y^\star$

Initialize Π with p_0; $\mathcal{S} \leftarrow \emptyset$; $q^\star \leftarrow -\infty$
while Π continues candidate search **do**
  $\mathcal{C} \leftarrow \Pi.\,\mathrm{NextRound}()$
  for each $(p_i, s_i) \in \mathcal{C}$ in predefined order **do**
    $(z_i^{(1)}, \omega_i, \hat{z}_{0,i}) \leftarrow \mathrm{FullWarmUp}(p_i, s_i)$
    if $\mathcal{S} = \emptyset$ **then**
      $\mathcal{S} \leftarrow \mathrm{SelectK}(\hat{z}_{0,i}, K)$ using Eq. (4)
    end if
    Save candidate-specific $(z_i^{(1)}, \omega_i)$
    $\tilde{z}_i^{(1)} \leftarrow \mathrm{Pack}(z_i^{(1)}|_{\mathcal{S}})$ in temporal order
    for $d = 2, \ldots, D$ **do**
      $\tilde{z}_i^{(d)} \leftarrow \mathrm{SparseStep}(\tilde{z}_i^{(d-1)}, p_i, d)$ with temporal RoPE $0{:}K{-}1$
    end for
    $\tilde{y}_i \leftarrow \mathrm{VAEDecode}(\tilde{z}_i^{(D)})$; $q_i \leftarrow V(\tilde{y}_i, p_0)$
    if $q_i > q^\star$ **then**
      $q^\star \leftarrow q_i$; $j \leftarrow i$
    end if
  end for
  $\Pi.\,\mathrm{Update}(\{\tilde{y}_i, q_i\}_{i \in \mathcal{C}})$ after the round
end while
Restore the selected candidate's $(z_j^{(1)}, \omega_j)$
for $d = 2, \ldots, D$ **do**
  $z_j^{(d)} \leftarrow \mathrm{FullStep}(z_j^{(d-1)}, p_j, d; \omega_j)$ over all F positions
end for
return $y^\star \leftarrow \mathrm{VAEDecode}(z_j^{(D)})$

Input and generation settings. All few-step models run in text-to-video mode with CFG scale 1.0. MiniMax-H3 uses the FL2VA backbone with the Turbo 8-step v1.0 768p adapter and a FlowMatch scheduler, with video and audio flow shifts of 6.0 and 3.0, respectively. HunyuanVideo-1.5 8B uses the LightX2V 8-step adapter with Euler flow matching and flow shift 3.0. Wan2.1-T2V-14B uses the LightX2V 4-step adapter with DMD2 Euler updates, flow shift 5.0, and pre-shift timesteps $(1000, 750, 500, 250)$. Full latent temporal lengths are 37, 31, and 21 for H3, Hunyuan, and Wan, respectively. For the 50-step baselines, we remove the few-step adapters and use the corresponding base checkpoints. H3 uses FlowMatch Euler sampling with CFG 1.0 and video/audio flow shifts of 12.0/3.0; Hunyuan uses Euler flow matching with CFG 6.0 and flow shift 5.0; Wan uses UniPC sampling with CFG 5.0 and flow shift 5.0.

Sparse preview decoding. The selected latent positions are sorted in temporal order, packed into a compact sequence, and jointly decoded by the model's native temporal VAE without reconstructing omitted positions. H3 requires at least three latent positions and decodes temporal chunks of five positions with an overlap of two. The final chunk is edge-padded only for alignment, and padded outputs are removed after decoding; spatial decoding uses tiles of size 256 with overlap 64. Hunyuan and Wan map K retained latent positions to $4K - 3$ RGB frames. At the selected budgets, Hunyuan uses spatial tiles of size 256 with 25% overlap and no temporal tiling, while Wan uses sequential causal decoding with a per-video feature cache and no temporal padding. The resulting sparse previews contain 56, 57, and 33 RGB frames for H3, Hunyuan, and Wan, respectively.

Search and evaluation settings. The latent temporal budgets are fixed at $K = 17$, 15, and 9 for H3, Hunyuan, and Wan. The Planner, Verifier, and Analyzer all use GPT-5.5. The paired ranking analysis uses eight seeds per prompt and VQA questions generated from the original prompt, following Section 3.1. Sparse previews and full videos use the same scoring protocol.

CachedSearch. We adapt CachedSearch (Saini et al., 2026) to the few-step samplers by retaining one fully computed warm-up step and one fully computed cooldown step, compared with five steps at each end of the original 50-step schedule. The remaining steps follow its accumulated-drift caching rule with $\tau = 0.1$. For H3, Hunyuan, and Wan, this leaves 6, 6, and 2 cache-eligible steps, respectively. The recorded timing runs skip 2, 2, and 1 DiT evaluations per candidate, yielding 6, 6, and 3 executed evaluations. Cache states are reset between candidates. After selection, the winning prompt–seed pair is regenerated using the corresponding few-step sampler with caching disabled.

SVG2 configuration. We use SVG2 (Yang et al., 2025) with 200 query centroids, 1000 key centroids, a centroid Top-p threshold of 0.9, and a minimum key-centroid ratio of 0.1. K-means uses up to 50 iterations for initialization and two iterations for subsequent updates, with centroids reused across timesteps and reset between videos. All layers use dense attention at the first denoising timestep; the first transformer layer also remains dense at subsequent timesteps. The mean retained attention densities within sparse calls are approximately 20.6%, 26.2%, and 30.6% for H3, Hunyuan, and Wan, respectively. Wan uses the upstream SVG2 attention processor, while H3 and HunyuanVideo-1.5 use architecture-specific adapters that preserve dense attention connections involving conditioning tokens.

SVG2 implementation. The initial Triton-based adaptation incurred substantial kernel and synchronization overhead, with denoising latency approximately twice that of dense inference in the Hunyuan timing runs. Following the execution path used for Wan, we adopt FlashInfer variable-block sparse attention, reuse device workspaces and token-layout metadata, and defer density-statistic transfers until the end of each rollout. These changes retain the same clustering parameters and sparse-attention schedule. For Hunyuan, the mean denoising latency decreases from 69.3 s to 32.7 s, giving a 1.14× speedup over the paired dense reference of 37.1 s. The optimized H3 implementation similarly achieves a 1.16× denoising speedup. We use these optimized implementations for the reported comparisons.

Agent prompts and search control. Appendix H provides the complete prompt templates. For each original prompt, the Planner is called once to produce a frozen semantic plan, 5–10 weighted atomic criteria, and four semantically equivalent generation prompts. The plan and criteria remain unchanged throughout search. The Verifier evaluates each candidate from an ordered, endpoint-inclusive 11-frame storyboard using only the original prompt and the frozen criteria, and computes the candidate score as their weighted average. After every candidate-search round, the Analyzer uses the globally best candidate's storyboard, criterion-level evidence, and search history to decide whether further refinement is useful. A `refine` decision produces four complete prompt variants for the next candidate-search round, whereas `reject` stops further refinement and sends the current globally best prompt–seed pair to final generation without another rewrite. Search is also bounded by a configured maximum number of candidate-search rounds; if this limit is reached before

`reject`, the current globally best pair proceeds to final generation. The final round denotes the single full-length video generation stage rather than another candidate-search round. Full-Agent, Cached-Agent, and Sparse-Agent use the same Analyzer policy and round limit, but may realize different mean candidate-search round counts because their candidate evidence differs.

B Benchmark Score Breakdowns

Table B.1: T2V-CompBench category scores for CachedSearch and Sparse Evolution under best-of-N search (scaled by 100; higher is better). Consist. and Dynamic denote attribute binding, Interact. denotes object interactions, and AVG is the seven-category mean.

Model	Method	Consist.	Dynamic	Spatial	Motion	Action	Interact.	Numeracy	AVG
H3	CachedSearch	80.6	27.4	62.3	36.8	69.7	84.1	37.4	56.9
	Sparse Evolution	80.5	27.0	62.4	36.4	69.5	84.0	37.1	56.7
Hunyuan	CachedSearch	76.8	23.9	57.1	31.4	63.8	79.6	35.6	52.6
	Sparse Evolution	76.7	23.8	57.2	31.1	63.7	79.5	35.5	52.5
Wan	CachedSearch	77.8	24.6	59.8	34.5	66.9	81.7	36.2	54.5
	Sparse Evolution	77.9	24.8	60.0	34.9	67.1	81.8	36.4	54.7

Table B.2: VBench Semantic scores for the same methods as Table B.1 (scaled by 100; higher is better). The five dimensions are object class, multiple objects, human action, color, and spatial relationship; Semantic is the official overall score.

Model	Method	Object	Multiple	Action	Color	Spatial	Semantic
H3	CachedSearch	91.4	73.8	96.2	84.5	58.6	77.7
	Sparse Evolution	91.3	73.5	96.3	84.4	58.1	77.6
Hunyuan	CachedSearch	78.6	43.8	91.2	71.5	34.7	64.7
	Sparse Evolution	78.7	43.6	91.3	71.4	34.8	64.7
Wan	CachedSearch	90.2	74.6	95.8	82.7	59.4	77.2
	Sparse Evolution	90.3	74.9	95.7	82.9	59.8	77.3

C Search Timing

Measured generation timing. The full and sparse generation latencies in Tables 1 and 4 are measured directly with generator weights resident in GPU memory. Per-candidate timing starts when candidate generation is invoked and ends when its decoded video is available. Per-prompt generation timing covers candidate exploration and completion of the selected final video. The measured interval includes the full first step, subsequent dense or sparse denoising, frame selection or caching when applicable, and VAE decoding. Model loading, offline benchmark evaluation, and one-time environment initialization are excluded. For H3, Hunyuan, and Wan, the retained budgets are $K = 17$, 15, and 9, respectively.

C.1 End-to-End Search Timing

For best-of-N search and iterative prompt refinement comparisons, wall-clock latency is measured directly from submission of a prompt-level search request until the selected final video is decoded, with generator weights already resident in GPU memory. It covers candidate generation and decoding, verifier and agent calls, search control, and final-video completion. Offline benchmark scoring, model loading, and one-time environment initialization are excluded. Methods share the same evaluation prompts, hardware, model residency, concurrency, and search stopping rules within each comparison.

Tables C.1 and C.2 provide stage breakdowns for the same measured requests. Gen., Cand., Verifier, Agent, and Final identify time spent in the corresponding stages; the directly

measured Wall-clock column is not reconstructed from the displayed rounded components. The selected Full-Agent candidate is already full-length and therefore incurs no additional Final stage.

Original VQQA produces full candidates and needs no separate final completion; VQQA + Sparse completes the selected video from its saved warm-up state.

Speedup divides the baseline Wall-clock by the method Wall-clock, using Best-of-8, Full-Agent, and VQQA as baselines for their respective search settings. Best-of-N search uses $N = 8$ candidates and zero prompt-refinement rounds.

Table C.1: End-to-end search time and stage breakdown on T2V-CompBench (seconds per prompt).

Model	Method	Gen.	Cand.	Verifier	Agent	Final	Wall-clock↓	Speedup
(a) Best-of-N search								
H3	Best-of-8	628.0	628.0	1.5	0.0	0.0	629.5	1.00×
	CachedSearch	570.9	492.4	1.5	0.0	78.5	572.4	1.10×
	Sparse Evolution	327.4	257.7	1.5	0.0	69.7	328.9	1.91×
Hunyuan	Best-of-8	379.2	379.2	1.5	0.0	0.0	380.7	1.00×
	CachedSearch	354.4	307.0	1.5	0.0	47.4	355.9	1.07×
	Sparse Evolution	217.1	174.6	1.5	0.0	42.5	218.6	1.74×
Wan	Best-of-8	175.2	175.2	1.5	0.0	0.0	176.7	1.00×
	CachedSearch	157.8	135.9	1.5	0.0	21.9	159.3	1.11×
	Sparse Evolution	98.9	81.7	1.5	0.0	17.2	100.4	1.76×
(b) Iterative prompt refinement (our agent)								
H3	Full-Agent	543.3	543.3	2.4	4.6	0.0	550.3	1.00×
	Cached-Agent	493.9	415.4	2.5	4.7	78.5	501.1	1.10×
	Sparse-Agent	292.1	222.4	2.6	4.8	69.7	299.5	1.84×
Hunyuan	Full-Agent	328.9	328.9	2.4	4.6	0.0	335.9	1.00×
	Cached-Agent	307.4	260.0	2.5	4.7	47.4	314.6	1.07×
	Sparse-Agent	193.5	151.0	2.6	4.8	42.5	200.9	1.67×
Wan	Full-Agent	151.5	151.5	2.4	4.6	0.0	158.5	1.00×
	Cached-Agent	136.5	114.6	2.5	4.7	21.9	143.7	1.10×
	Sparse-Agent	87.6	70.4	2.6	4.8	17.2	95.0	1.67×
(c) Iterative prompt refinement (VQQA)								
H3	VQQA	212.0	212.0	4.1	9.4	0.0	225.5	1.00×
	VQQA + Sparse	152.5	82.8	4.1	9.4	69.7	166.0	1.36×
Hunyuan	VQQA	156.4	156.4	5.0	11.5	0.0	172.9	1.00×
	VQQA + Sparse	104.3	61.8	5.0	11.5	42.5	120.8	1.43×
Wan	VQQA	61.3	61.3	4.2	9.8	0.0	75.3	1.00×
	VQQA + Sparse	41.2	24.0	4.2	9.8	17.2	55.2	1.36×

Table C.2: End-to-end search time and stage breakdown on VBench Semantic (seconds per prompt).

Model	Method	Gen.	Cand.	Verifier	Agent	Final	Wall-clock↓	Speedup
(a) Best-of-N search								
H3	Best-of-8	628.0	628.0	1.5	0.0	0.0	629.5	1.00×
	CachedSearch	570.9	492.4	1.5	0.0	78.5	572.4	1.10×
	Sparse Evolution	327.4	257.7	1.5	0.0	69.7	328.9	1.91×
Hunyuan	Best-of-8	379.2	379.2	1.5	0.0	0.0	380.7	1.00×
	CachedSearch	354.4	307.0	1.5	0.0	47.4	355.9	1.07×
	Sparse Evolution	217.1	174.6	1.5	0.0	42.5	218.6	1.74×
Wan	Best-of-8	175.2	175.2	1.5	0.0	0.0	176.7	1.00×
	CachedSearch	157.8	135.9	1.5	0.0	21.9	159.3	1.11×
	Sparse Evolution	98.9	81.7	1.5	0.0	17.2	100.4	1.76×
(b) Iterative prompt refinement (our agent)								
H3	Full-Agent	413.1	413.1	1.9	3.6	0.0	418.6	1.00×
	Cached-Agent	379.0	300.5	1.9	3.7	78.5	384.6	1.09×
	Sparse-Agent	238.8	169.1	2.0	3.8	69.7	244.6	1.71×
Hunyuan	Full-Agent	247.6	247.6	1.9	3.6	0.0	253.1	1.00×
	Cached-Agent	233.6	186.2	1.9	3.7	47.4	239.2	1.06×
	Sparse-Agent	156.7	114.2	2.0	3.8	42.5	162.5	1.56×
Wan	Full-Agent	115.2	115.2	1.9	3.6	0.0	120.7	1.00×
	Cached-Agent	102.8	80.9	1.9	3.7	21.9	108.4	1.11×
	Sparse-Agent	70.6	53.4	2.0	3.8	17.2	76.4	1.58×
(c) Iterative prompt refinement (VQQA)								
H3	VQQA	196.3	196.3	3.8	8.7	0.0	208.8	1.00×
	VQQA + Sparse	146.5	76.8	3.8	8.7	69.7	159.0	1.31×
Hunyuan	VQQA	146.9	146.9	4.7	10.8	0.0	162.4	1.00×
	VQQA + Sparse	100.6	58.1	4.7	10.8	42.5	116.1	1.40×
Wan	VQQA	56.9	56.9	3.9	9.1	0.0	69.9	1.00×
	VQQA + Sparse	39.5	22.3	3.9	9.1	17.2	52.5	1.33×

D Cost Analysis

Following CachedSearch (Saini et al., 2026), we account for candidate exploration and final generation separately. Its regeneration setting requires a complete generation pass after candidate exploration. Sparse Evolution instead reuses the selected candidate's full-sequence warm-up state and performs the remaining denoising steps.

This analysis considers denoising compute in the regime where video generation dominates the cost. Let C_F denote the cost of denoising a full video and C_W the cost of its first full-sequence warm-up step. Search explores n prompt–seed candidates. The sparse exploration cost $C_{S,i}$ of candidate i includes full-sequence warm-up and subsequent denoising over the selected K temporal positions. With $\bar{C}_S$ denoting the average candidate cost,

$$C_{\mathrm{SE}} = \underbrace{\sum_{i=1}^{n} C_{S,i}}_{\text{sparse exploration}} + \underbrace{(C_F - C_W)}_{\text{full-video completion}} = n\bar{C}_S + C_F - C_W. \tag{5}$$

If full-sequence denoising steps have approximately equal cost, $C_W \approx C_F/D$, giving

$$\boxed{C_{\mathrm{SE}} \approx n\bar{C}_S + \frac{D-1}{D} C_F} \tag{6}$$

The first term accounts for the candidates generated during search, and the second accounts for the remaining denoising needed to complete the final video. The candidate count n is the actual number explored before search terminates and may vary across requests.

Generating all n candidates at full temporal length has a reference cost of $C_{\text{full}} = nC_F$. With $\gamma = \bar{C}_S/C_F$, the denoising speedup of Sparse Evolution relative to this reference is

$$\text{Speedup} \approx \frac{nC_F}{n\bar{C}_S + \frac{D-1}{D}C_F} = \frac{n}{n\gamma + \frac{D-1}{D}}. \tag{7}$$

When $n(1-\gamma) > (D-1)/D$, the computation saved during candidate exploration exceeds the additional cost of final-video completion. As the number of explored candidates increases, this completion cost is amortized over more candidates, and the overall benefit is governed primarily by the cost ratio between sparse and full candidates.

Keyframe previews reduce denoising computation during candidate exploration, while warm-up reuse removes repeated computation during final-video completion. The reported generation times include both preview decoding and final full-video decoding.

E Temporal Selection at the Same Latent Temporal Budget

We compare Random-K, Uniform-K, and Adaptive-K while fixing the number of retained latent temporal positions. The budgets are $K = 17$, 15, and 9 for H3, Hunyuan, and Wan, respectively. Ranking and selection use the same 140 prompts and eight seeds per prompt as Table 4; these prompts are excluded from the final benchmark evaluation. The strategies share warm-up states, denoising steps, temporal RoPE handling, VAE decoding, and verifier protocol. Candidate time follows the same timing protocol as Table 4.

Selection rules. Uniform-K retains exactly K evenly spaced, distinct temporal positions, including the first and last positions. Random-K retains the first position and samples the remaining $K-1$ positions uniformly without replacement. Adaptive-K follows Section 2.3.1. Each strategy selects one position set per prompt, sorted in temporal order and shared by its eight candidates; Random-K samples its set once. The every-other-position Uniform baseline in Table 4 retains more positions and is distinct from Uniform-K here.

Metrics. Table E.1 reports the full-video verifier score of the selected seed, scaled by 100, together with mean per-prompt Spearman correlation, Top-1 agreement, and Regret as defined in Section 2.2. These scores use the verifier protocol rather than the official benchmark metrics. Candidate time includes warm-up, selection, denoising, and decoding, so the selection overhead remains part of the comparison.

Table E.1: Temporal selection at the same latent temporal budget. Score is the mean full-video verifier score of the selected seeds, scaled by 100. Time is in s/candidate, including frame selection and VAE decoding; Top-1 and gain capture are percentages. Adaptive-K reproduces the matching values from Table 4.

Model	Selection	K	Score ↑	ρ_s ↑	Top-1 ↑	Regret ↓	Gain capture ↑	Time ↓
H3	Random-K	17	67.88	0.70	56	0.0327	70.6	32.21
	Uniform-K		68.43	0.73	59	0.0272	75.5	32.21
	Adaptive-K		69.86	0.78	62	0.0129	88.4	32.21
Hunyuan	Random-K	15	66.28	0.67	52	0.0363	63.3	21.82
	Uniform-K		66.86	0.70	54	0.0305	69.2	21.82
	Adaptive-K		68.10	0.75	59	0.0181	81.7	21.82
Wan	Random-K	9	67.06	0.72	57	0.0315	69.0	10.21
	Uniform-K		67.44	0.74	59	0.0277	72.7	10.21
	Adaptive-K		68.94	0.81	63	0.0127	87.5	10.21

F Ablation of Sparse-Generation Components

The component ablations use the prompts, candidate seeds, model-specific K, and verifier metrics of Appendix E. Table F.1 separates the effect of the initial full-sequence step from temporal RoPE re-indexing. The selection-strategy comparison is reported in Table E.1.

Full-sequence warm-up. We fix the same Adaptive-K indices from the paired full-first-step run in both variants, isolating the effect of the warm-up step. The full-first-step variant performs one full-sequence denoising step followed by $D-1$ sparse steps. The sparse-first-step variant starts from the corresponding selected positions of the same initial noise and performs all D steps sparsely. Both variants use the same total number of denoising steps, RoPE convention, and decoder. Their sparse scores are compared against the same full-video candidate references. A sparse first step saves time but sharply reduces verifier scores and ranking correlations, making full-sequence warm-up necessary.

Temporal RoPE indices. We fix the Adaptive-K selections and full-sequence warm-up states, and vary only the temporal position indices used during sparse denoising. Original indices preserve the selected positions in the full sequence; compact indices use $0, \ldots, K-1$, as in the main method. Spatial position indices remain unchanged.

Table F.1: Ablation of the initial full-sequence step and temporal RoPE indices. Both comparisons use the same Adaptive-K positions; RoPE variants also share warm-up states. Scores and budgets follow Table E.1; time is in s/candidate.

Model	Component	Variant	Score ↑	ρ_s ↑	Regret ↓	Time ↓
H3	Warm-up	**Full first step**	69.86	0.78	0.0129	32.21
		Sparse first step	65.35	0.37	0.0580	26.05
	RoPE	Original indices	68.46	0.71	0.0269	32.21
		Compact indices	69.86	0.78	0.0129	32.21
Hunyuan	Warm-up	**Full first step**	68.10	0.75	0.0181	21.82
		Sparse first step	61.27	0.33	0.0864	18.92
	RoPE	Original indices	67.53	0.72	0.0238	21.82
		Compact indices	68.10	0.75	0.0181	21.82
Wan	Warm-up	**Full first step**	68.94	0.81	0.0127	10.21
		Sparse first step	63.09	0.29	0.0712	6.97
	RoPE	Original indices	68.12	0.77	0.0209	10.21
		Compact indices	68.94	0.81	0.0127	10.21

Reuse of the selected warm-up state. For the same selected prompt–seed pair, Table F.2 compares full-video generation restarted from its original initialization with completion from the saved full-sequence latent and sampler state. Both paths use the same generator, sampling schedule, and VAE. Completion time includes denoising and final decoding. The absolute difference between the two full-video verifier scores is reported on the same $\times 100$ scale.

Table F.2: Final-video completion with and without reuse of the selected full-sequence warm-up state. Restart and resume use the same selected prompt–seed pair. Times are in s/video and include VAE decoding; speedup is restart time divided by resume time.

Model	Restart	Resume	Speedup ↑	Absolute score gap ↓
H3	78.50	69.71	1.13×	0.00
Hunyuan	47.40	42.52	1.11×	0.00
Wan	21.90	17.23	1.27×	0.00

G Ablation of Evidence Feedback in Iterative Prompt Refinement

We conduct a 2×2 ablation to examine whether generation-derived evidence improves iterative prompt refinement and whether sparse previews preserve this benefit. We vary the candidate-generation backend (Full or Sparse) and the availability of evidence feedback to the prompt rewriter. All four conditions retain prompt refinement and verifier-based candidate selection, isolating the contribution of generation feedback rather than the benefits of prompt variation or candidate search itself.

Experimental setup. We use the same three generators and T2V-CompBench evaluation prompts as in Section 3.3. For each original prompt, all four conditions share the initial Planner output, frozen semantic plan, evaluation criteria and weights, and four initial prompt variants. Generation and verification settings follow Appendices A and H. Each condition runs exactly two candidate-search rounds with four candidates per round, exploring eight candidates in total and performing one prompt-refinement call between rounds. Early stopping is disabled only for this ablation. Seeds are assigned in advance to each round and candidate slot to prevent differences in search budget from confounding the feedback comparison.

With evidence feedback. After the first round, the Analyzer receives the highest-scoring candidate's ordered storyboard, criterion-level scores, visual evidence, and search history. It uses this information to generate four complete, semantics-preserving prompts for the next round. Evidence comes from full videos or sparse previews according to the backend, while refinement rules and semantic constraints remain unchanged.

Without evidence feedback. The same language model generates four next-round prompt variants using only the original prompt, frozen semantic plan and evaluation criteria, and a chronological history of previous prompts. The rewriter receives no candidate images, scores, rankings, or generation-derived failure diagnoses. The `current_best_candidate` field is removed, and prompt history is neither selected nor ordered by score. The Verifier still scores all candidates and determines the final selection, but its outputs are not passed to the rewriter. This condition therefore preserves the number of refinement calls, candidate budget, and semantic constraints while removing generation-derived guidance from subsequent prompt refinement.

Candidate selection and evaluation. After both rounds, each condition selects the highest-scoring prompt–seed pair among all eight candidates using verifier scores conditioned on the original prompt p_0. Full returns the selected full video, whereas Sparse completes the selected candidate from its saved full-sequence warm-up state. Final outputs are evaluated using the seven-category mean of T2V-CompBench, scaled by 100, consistent with the main experiments rather than using search-time verifier scores.

Table G.1: Evidence-feedback ablation under a fixed candidate budget. All conditions perform iterative prompt refinement and verifier-based candidate selection, varying only the candidate backend and the availability of generation-derived evidence to the rewriter. Entries are final-video T2V-CompBench scores. Feedback gain is the score difference between the with-feedback and without-feedback conditions, measured in points.

Model	Backend	Without feedback	With feedback	Feedback gain Δ_b
H3	Full	56.1	60.5	+4.4
	Sparse	55.8	59.6	+3.8
Hunyuan	Full	51.9	57.5	+5.6
	Sparse	52.0	57.3	+5.3
Wan	Full	55.6	59.4	+3.8
	Sparse	55.3	58.8	+3.5

Results. For candidate backend b, we define the incremental benefit of evidence feedback as

$$\Delta_b = Q_{b,\text{with}} - Q_{b,\text{without}}, \qquad b \in \{\text{Full}, \text{Sparse}\}, \tag{8}$$

where Q denotes the final-video T2V-CompBench score.

As shown in Table G.1, evidence feedback improves final scores across all three generators under both candidate backends. Full achieves feedback gains of 4.4, 5.6, and 3.8 points, while Sparse achieves gains of 3.8, 5.3, and 3.5 points, respectively. Relative to the Full backend for each model, Sparse retains 86.4%, 94.6%, and 92.1% of the incremental feedback gains. These results show that evidence-guided refinement outperforms prompt variation without generation-derived evidence, and that sparse candidates preserve most of this benefit. Beyond supporting candidate selection, sparse previews retain semantic evidence that can guide subsequent prompt refinement and provide useful feedback for iterative search.

H Agent Prompt Templates

The Planner, Verifier, and Analyzer use GPT-5.5. Their complete system and user prompts are reproduced below, with runtime placeholders retained in braces.

H.1 Planner and Initial Prompt Rewrite

System prompt.

```
You are a precise text-to-video semantic planning and prompt-optimization expert. Return
exactly one valid JSON object and no markdown.
```

User prompt.

```
## Role

You are an expert text-to-video semantic planning and prompt-optimization
specialist. Starting only from the immutable original prompt, create:

1. a frozen semantic verification plan;
2. 5-10 weighted atomic semantic criteria; and
3. exactly {candidate_budget} complete, semantically equivalent generation
   prompt variants.

The semantic plan and criteria are generated once and then reused unchanged for
every round, seed, candidate, and final full video associated with this original
prompt.

Original prompt:
{original_prompt}

Candidate budget:
{candidate_budget}

## Semantic planning rules

- Recover every explicit entity, count, attribute, binding, spatial relation,
  action, interaction, motion direction, initial state, final state, and state
  transition in the original prompt.
- Preserve the original semantics. Do not add new objects, colors, materials,
  anatomy, locations, actions, outcomes, event ordering, narrative beats,
  visual styles, lighting conditions, or camera movements.
- List likely semantic failure modes, but do not convert those risks into new
  generation requirements.

## Frozen semantic criteria

- Create 5-10 simple, independently answerable criteria. Each criterion must
```

```
  test exactly one visible semantic fact.
- Together, the criteria must cover every explicit requirement in the original
  prompt.
- Prefer atomic questions over compound questions. Separate facts that can fail
  independently.
- Presence and setting questions may be used as low-weight prerequisites when
  they make later failures easier to diagnose.
- Do not manufacture question count by paraphrasing the same fact. If the same
  storyboard evidence would answer two questions identically, merge them.
- An action criterion naming the actor, action, target, and direction already
  checks action binding; do not add a redundant binding criterion.
- Put most of the weight on the compositional operation that defines success:
  the requested count, binding, relation, action, interaction, direction, or
  state change. Core-operation criteria should normally receive at least 55%
  of the total raw weight. Presence and context prerequisites should normally
  receive no more than 25%.
- For an action, distinguish the requested action from standing, posing, or an
  easier substitute.
- For directional motion, require visible subject displacement in the stated
  direction relative to the stated reference. Limb motion or camera motion is
  not sufficient.
- For temporal change, test the ordered initial state, final state, and the
  requested transition. Uniformly sampled frames do not need to show every
  instant.
- For interaction, require the specified entities to participate visibly, but
  do not invent an unstated body part, contact point, or intermediate pose.
- For numeracy, test the requested count and require distinct, non-merged
  entities.
- Accept semantically equivalent visual realizations. Do not introduce hidden
  assumptions.
- Never create questions about sharpness, beauty, cinematic quality, lighting,
  temporal smoothness, deformation, compression, or visual artifacts.
- Use five questions for a simple prompt, usually six to eight for a moderately
  composite prompt, and nine to ten only when necessary.
- Every question must begin exactly with: "On a scale of 0-100, how"
- Every weight must be positive. The weights must sum to 1.0.
- Before returning, conduct a pairwise redundancy audit and remove criteria
  whose evidence and answers would be essentially identical.

## Generation prompt variants

- Return exactly {candidate_budget} distinct, non-empty English prompts.
- Each variant must be a complete prompt that can be sent directly to the video
  generator. Do not return fragments or editing instructions.
- Make every required subject, binding, relation, action, direction, and state
  transition visually executable and easy to observe.
- For a temporal prompt, express the required initial state, transition, and
  final state in a coherent temporal sequence.
- Rewrite the entire scene coherently. Do not append a checklist of warnings or
  generic negative phrases.
- Variants may differ in wording and neutral visibility-aiding composition.
  A fixed, wide, or side view may be used only when it helps expose an explicit
  count, relation, action, path, interaction, or state change.
- Do not add generic padding such as "cinematic," "photorealistic," "4K,"
  "high quality," "sharp," or "smooth."
- Do not add evaluator-facing or benchmark-facing instructions.
- All variants must remain semantically equivalent to the original prompt.

Return exactly one JSON object:
{
  "original_prompt": "copy the original prompt exactly",
  "semantic_plan": {
    "entities": [],
    "bindings": [],
    "spatial_relations": [],
    "motion_events": [],
    "state_changes": [],
    "failure_risks": []
```

```
  },
  "must_preserve": [
    "one plain-language statement for each immutable original requirement"
  ],
  "prompt_variants": [
    "exactly candidate_budget complete English generation prompts"
  ],
  "criteria": [
    {
      "id": "action_1",
      "type": "action_binding",
      "requirement": "one visible atomic semantic fact",
      "question": "On a scale of 0-100, how clearly ...?",
      "weight": 0.0
    }
  ],
  "notes": "Frozen semantic generation and verification plan"
}
```

H.2 Semantic Verifier

System prompt.

```
You are a precise text-to-video semantic alignment evaluator. Use only the original
prompt, frozen criteria, and attached storyboard. Return exactly one valid JSON object and
no markdown.
```

User prompt.

```
## Role

You are an expert text-to-video semantic fidelity evaluator. Judge only the
compositional semantic fidelity of the candidate video to the immutable
original prompt. The attached storyboard contains 11 ordered frames numbered
0 through 10 from one candidate video.

Original prompt:
{original_prompt}

Frozen semantic criteria generated only from the original prompt:
{semantic_criteria}

## Evaluation rules

- Answer every frozen criterion exactly once and in the supplied order.
- Score only the semantic requirement stated by the current criterion.
- Treat each criterion as an atomic diagnostic. Provide criterion-specific
  visible evidence instead of repeating one generic explanation.
- Do not inflate an entity-presence or setting score merely because the scene
  is plausible. Do not award an action, relation, or interaction merely because
  its entities are present.
- Do not lower semantic scores for blur, sampled-frame jumps, frame rate,
  visual polish, lighting, cinematic quality, compression, or minor artifacts.
  These are not visual-quality questions. Mention an artifact only when it
  genuinely prevents the required semantic fact from being judged.
- Penalize missing or wrong entities, bindings, counts, spatial relations,
  actions, interactions, motion directions, and ordered state transitions.
- For an action, require visible evidence of the requested action rather than a
  static pose or a semantically different movement.
- For directional motion, compare subject position across ordered frames and
  distinguish object displacement from camera motion.
- For temporal change, require distinguishable initial and final states in the
  correct order. Uniformly sampled intermediate frames are acceptable.
- For an interaction, require visible participation by the specified entities,
```

```
  but never require an unstated body part, exact contact point, camera view, or
  intermediate pose.
- Accept any semantically valid realization of the original prompt.
- `evidence_frames` must contain zero-based storyboard frame indices.
- `confidence` is confidence in the semantic judgment, not image quality.

Semantic scoring rubric:
- 0-20: absent, contradicted, or catastrophically wrong
- 21-40: weak evidence or a major semantic failure
- 41-60: partially fulfilled, with an important requirement missing
- 61-80: mostly fulfilled, with a minor semantic mismatch
- 81-99: clear and strongly fulfilled
- 100: unambiguous complete fulfillment; use sparingly

Compute the overall score as the frozen weighted average of `vqa_score`, divided
by 100. Do not add a global visual-quality score.

Return exactly one JSON object:
{
  "score": 0.0,
  "vqa_results": [
    {
      "criterion_id": "copy the criterion id",
      "question": "copy the criterion question exactly",
      "vqa_score": 0,
      "weight": 0.0,
      "reason": "brief visible semantic evidence",
      "evidence_frames": [0],
      "confidence": 0.0
    }
  ],
  "failure_reasons": ["only failed semantic requirements"],
  "evidence": "brief overall semantic evidence",
  "semantic_summary": "which original requirements are fulfilled or missing"
}
```

H.3 Evidence-Guided Analyzer and Prompt Rewrite

System prompt.

```
You are a precise visual-evidence-guided text-to-video prompt optimizer. Preserve the
immutable original semantics, diagnose only evidence-supported semantic failures, and
return exactly one valid JSON object with no markdown.
```

User prompt.

```
## Role

You are an expert visual-evidence-guided text-to-video prompt optimizer. The
attached storyboard belongs to the globally best candidate found so far. Use
its criterion-level semantic evidence and the search history first to decide
whether further prompt refinement is useful. If refinement should continue,
produce the complete generation prompts for the next candidate-search round.

Input JSON:
{analyzer_input_json}

The input contains:
- `original_prompt`: the immutable original prompt;
- `candidate_budget`: the required number of next-round prompts;
- `round_index`: the current refinement round;
- `frozen_semantic_plan`: the semantic structure generated once from the
  original prompt and then kept unchanged;
- `frozen_semantic_criteria`: the weighted VQA scoring criteria generated once
```

```
  from the original prompt and then kept unchanged;
- `must_preserve`: the original requirements that every rewrite must retain;
- `current_best_candidate`: its generation prompt, seed, overall score, and
  criterion-level VLM evidence;
- `round_history`: previous prompts, scores, and structured semantic evidence.

## Stage 1: decide

- Use `reject` only when candidate refinement should stop and the current
  globally best prompt and seed should proceed to final full-video generation:
  the semantic evidence is already sufficient, the remaining failures are not
  actionable through prompt rewriting, or the scores have plateaued.
- Use `refine` when important frozen criteria remain visibly weak and a clearer
  generation prompt can plausibly correct them.
- `reject` means rejecting further refinement, not rejecting the candidate or
  final video.
- Never choose `reject` merely because the candidate storyboard is blurry,
  visually rough, or lacks dense in-between frames.

## Analysis rules

- Diagnose the current best candidate using only the attached storyboard and
  the supplied criterion-level evidence.
- Prioritize low-scoring, high-weight, or repeatedly failed semantic criteria.
- Preserve requirements that already succeed. Do not repair one criterion by
  weakening another original requirement.
- Distinguish prompt ambiguity from stochastic generation failure and from
  insufficient evidence in the candidate storyboard.
- Treat the original prompt, frozen semantic plan, frozen criteria, and
  `must_preserve` list as the semantic boundary. The current generation prompt
  is editable context, not a source of new requirements.
- Do not modify the frozen questions or their weights.

## Stage 2: rewrite when refining

- If the decision is `refine`, produce exactly `candidate_budget` distinct,
  non-empty, complete English generation prompts.
- If the decision is `reject`, do not rewrite: return an empty `next_prompt`
  and an empty `next_prompts` list.
- Rewrite the whole scene coherently. Do not merely append a warning or a short
  patch to the previous prompt.
- Make the failed high-value semantics more executable and visually observable.
  Explicitly bind every required actor to its action, target, direction,
  relation, or state transition.
- For a required temporal transformation, clearly express its initial state,
  visible progression, and final state in the requested order.
- For a required action or motion, describe visible action evidence and, when
  required, displacement relative to the stated reference.
- For numeracy, keep the requested entities visually distinct.
- Convert failure evidence into positive scene instructions whenever possible.
  A direct contrast may be used when it clearly expresses a required state
  transition.
- Preserve every original entity, count, attribute, relation, action,
  interaction, direction, and state transition.
- Do not invent objects, colors, materials, anatomy, contact mechanics,
  locations, story events, visual styles, lighting, or camera motion.
- A fixed, wide, or side view may be used only when it neutrally exposes an
  explicit count, relation, action, motion path, interaction, or state change.
- Do not add generic cinematic, photorealistic, 4K, high-quality, sharpness, or
  smoothness padding.
- Do not write evaluator-facing instructions such as "satisfy criterion 3,"
  "increase the score," or "pass the benchmark."
- Variants may use different semantics-preserving wording or neutral
  visibility-aiding composition, but must not change the requested event.

Return exactly one JSON object:
{
  "decision": "refine or reject",
```

```
  "reason": "evidence-grounded decision",
  "analysis": {
    "failed_criteria": [
      {
        "criterion_id": "criterion id",
        "observed_evidence": "what is visibly missing or wrong",
        "diagnosis": "why the current prompt may under-express the requirement",
        "rewrite_target": "how to express it more clearly without adding semantics"
      }
    ],
    "preserved_successes": [
      "successful requirements that must not be disturbed"
    ],
    "repeated_failures": [
      "semantic failures recurring across previous rounds"
    ]
  },
  "failure_focus": [
    "the most important unresolved semantic failures"
  ],
  "rewrite_hints": [
    "concrete semantic corrections, or an empty list on reject"
  ],
  "refinement_strategy": "the coherent next-round strategy, or empty on reject",
  "next_prompt": "the canonical complete rewrite, or empty on reject",
  "next_prompts": [
    "exactly candidate_budget complete prompts, or an empty list on reject"
  ]
}
```