CODED COMPUTATION: PERCEPTUAL-SEMANTIC ACCELERATION FOR VIDEO DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Efficient video diffusion is commonly approached by making generative computation cheaper through fewer denoising steps, quantized operators, feature reuse, or selective computation. We explore a different perspective—*can computation itself be coded and subsequently reconstructed?* Inspired by coded acquisition in computational imaging, we introduce *coded computation*, a new formulation that jointly designs which generative computations are explicitly performed and how the omitted information is recovered. We instantiate this principle with a perceptual-semantic computational encoder and a lightweight code-aware generative decoder. The encoder combines bottom-up visual cues with top-down semantic relevance from the text prompt to construct a time-varying compute code, allocating richer computation to perceptually and semantically important content while encoding less critical content more efficiently. The decoder then leverages the compute code and strong spatiotemporal and diffusion priors to reconstruct the omitted generative information. Extensive experiments demonstrate that coded computation accelerates video diffusion inference by 1.42–$2.64\times$ while preserving visual quality, temporal consistency, and prompt fidelity.

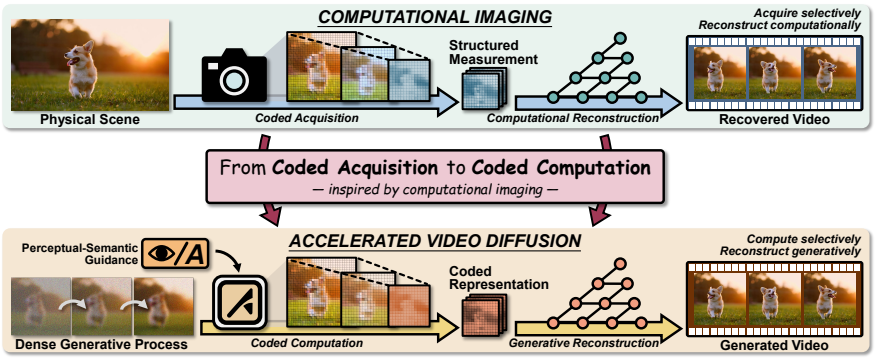

Figure 1: **From coded acquisition to coded computation.** Computational imaging selectively acquires structured measurements and recovers missing visual information via computational reconstruction. Analogously, we selectively execute perceptual-semantically guided generative computation to form a coded representation, from which the omitted generative information is reconstructed.

1 INTRODUCTION

Recent video diffusion models (Ma et al., 2025a; Kong et al., 2024; Team Wan et al., 2025; HaCohen et al., 2024) have achieved remarkable progress in visual fidelity, temporal coherence, and text alignment, but at a rapidly growing inference cost. Generating a video requires repeatedly evaluating large spatiotemporal diffusion transformers (DiTs) (Peebles & Xie, 2023; Yang et al., 2025b) over a long denoising trajectory, where spatiotemporal latent tokens are progressively refined from noise into structured scenes, motion, and visual details. As model size, spatial resolution, and video duration continue to increase, such iterative dense computation incurs substantial latency and computational overhead, severely limiting the scalability and accessibility of modern video generators (Zhang et al., 2025b). Reducing this inference cost while preserving generation

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.

quality has therefore become a central challenge for scaling video diffusion models toward higher resolutions, longer durations, and practical deployment.

Existing acceleration methods tackle this challenge from different perspectives. Step distillation reduces the number of model evaluations (Yin et al., 2025), quantization and efficient operators lower the cost of individual evaluations (Zhao et al., 2025a; Zhang et al., 2025a), caching exploits redundancy across correlated denoising steps (Kahatapitiya et al., 2025), and token- or block-level sparsification avoids computations predicted to be unnecessary (Li et al., 2026; Xi et al., 2025). These techniques have demonstrated substantial efficiency gains by exploiting the considerable redundancy present across spatial tokens, video frames, and denoising steps. Despite their different mechanisms, however, they largely share an *execution-centric* view of inference: computation is treated as a collection of operations to be executed more cheaply, approximated, reused, or removed. Even adaptive methods primarily focus on deciding which computations can be reduced or bypassed, rather than explicitly co-designing a structured computational code and a complementary reconstruction process. In other words, existing methods primarily optimize how computation is executed, but rarely formulate efficient inference as a joint problem of *encoding computation and decoding the generative information it represents*. This distinction motivates us to seek a different principle for efficient generative inference.

To move beyond this execution-centric view, we draw inspiration from computational imaging (Bhandari et al., 2022). Rather than densely and faithfully acquiring every signal sample, coded imaging systems deliberately redesign the acquisition process to capture structured, incomplete, yet informative measurements, and subsequently recover the desired signal through computational reconstruction (Levin et al., 2007; Martel et al., 2020; Metzler et al., 2020; Zhou et al., 2020). Their key insight is not merely to *measure less*, but to jointly design *what information should be acquired, how it should be encoded, and how the unmeasured information should be recovered*. Coded exposure and compressive imaging further demonstrate that the structure and diversity of measurements can be as important as their individual precision (Raskar et al., 2006; Vargas et al., 2021; Zhou et al., 2023; Gandikota et al., 2026), while foveated and task-aware imaging allocate limited sensing resources according to perceptual or semantic value (Zhao & Lam, 2024; Klotz & Nayar, 2024; Xiao et al., 2026). This acquisition–reconstruction paradigm suggests a fundamentally different way to think about generative inference. Video diffusion repeatedly refines a highly structured latent representation, with strong redundancy and priors spanning spatial locations, video time, semantics, and denoising steps. Rather than simply eliminating redundant computation, we therefore ask whether generative inference can be deliberately *coded*, such that only structured and informative computations are explicitly executed while the remaining generative information is recovered from strong generative priors. We refer to this new principle as *coded computation*, transferring the philosophy of coded acquisition and computational reconstruction from visual sensing to generative inference. As shown in Fig. 1, this establishes a direct analogy between coded sensing and coded generative computation.

Following this principle, we instantiate coded computation for video diffusion with a *perceptual-semantic computational encoder* and a lightweight *code-aware generative decoder*. After a short initial denoising stage establishes a coarse spatiotemporal scaffold, the encoder combines bottom-up visual cues, such as motion, structure, texture, and saliency, with top-down semantic relevance from the text prompt to construct a time-varying compute code. This code allocates richer DiT computation to perceptually and semantically important content, while representing less critical regions through substantially cheaper coded computation. Rather than simply discarding the omitted computation, the co-designed decoder exploits the compute code together with spatial, temporal, semantic, and diffusion-trajectory priors to reconstruct the missing generative information, allowing coded and fully computed content to evolve jointly within a unified latent representation.

Our experiments on Wan2.1-1.3B show that coded computation accelerates video generation by 1.42–$2.64\times$ while preserving high similarity to native outputs. Across these settings, SSIM ranges from 0.928 to 0.973, and VBench scores retain at least 99.7% of the native score, demonstrating that substantial computational savings can coexist with faithful video generation. Additional evaluation on LTX-Video-2B further supports the effectiveness of the framework across video generation backbones.

Our contributions are threefold:

- **Coded computation, a new computational-imaging-inspired paradigm for accelerating video diffusion.** It reformulates generative inference as a joint problem of computation encoding and reconstruction, rather than merely reducing or skipping existing operations.
- **A perceptual-semantic computational encoder for adaptive compute coding.** It combines bottom-up visual cues with top-down semantic relevance to determine where and how generative computation is allocated and encoded along the diffusion trajectory.
- **A code-aware generative decoder tailored to the encoded computation.** It leverages the compute code and strong spatiotemporal and diffusion priors to reconstruct generative information deliberately omitted during coded computation.

2 Related Work

Few-step distillation of video diffusion models. A major line of work accelerates video diffusion by compressing the original denoising trajectory into fewer model evaluations. Recent approaches improve few-step generation through stronger trajectory- or distribution-level supervision (Luo et al., 2025; Zheng et al., 2026; Zhang et al., 2026; Ge et al., 2026), stage-specific specialization for different noise regimes (Lv et al., 2025a; Fan et al., 2025), and alternative parameterizations of large denoising transitions (Nie et al., 2026; Gu et al., 2026; Shaul et al., 2026). These methods effectively learn a more efficient replacement for the teacher's sampling dynamics and can greatly reduce the number of denoising steps. Their main focus, however, is to shorten or redesign the sampling trajectory itself, whereas our work studies how computation is encoded and reconstructed within an evolving inference trajectory. The two directions are therefore complementary.

Computation reuse and selective execution. Another line of work reduces inference cost by exploiting redundancy across denoising steps, layers, and tokens. Feature caching methods reuse intermediate activations according to schedules (Lv et al., 2025b; Liu et al., 2025a; Ma et al., 2025c; Gao et al., 2025; Bu et al., 2026; Lv et al., 2026; Zhao et al., 2025b), while feature forecasting replaces skipped evaluations with inexpensive predictions derived from historical states (Liu et al., 2025b; Shen et al., 2026; Chen et al., 2026; Zou et al., 2026). More fine-grained approaches selectively allocate computation across blocks, tokens, attention interactions, or sampling stages based on estimated importance, sensitivity, or redundancy (Ma et al., 2025b; Zou et al., 2025; Liu et al., 2026; Yang et al., 2025a; Song et al., 2025; Chu & Patel, 2026). Despite their different execution granularities, these methods treat redundant computation as something to reuse, approximate, sparsify, or bypass. Conversely, our *coded computation* perspective treats selective execution and reconstruction as two coupled parts of the same inference process: executed computation forms a structured code, while a co-designed decoder explicitly recovers the generative information that is left uncomputed.

3 Method

3.1 Motivation and Formulation of Coded Computation

Problem setup. Let z_t denote the video latent at diffusion step t, σ_t its noise level, and c the text condition. We consider a coded segment from an entrance step x to a later exit step y, with $\sigma_x > \sigma_y$, and denote by $\mathcal{T}_{x\to y}$ the ordered sampler steps between them. The pretrained video diffusion transformer F_θ remains frozen. At the segment entrance, $\bar{z}_x$ denotes the clean-content estimate associated with the current sampling state, after an initial dense denoising stage has established a coarse spatiotemporal structure. We index the N spatiotemporal DiT tokens by $i \in \{1, \dots, N\}$.

Why can generative computation be coded? Our formulation draws inspiration from computational imaging, where incomplete but informative measurements are deliberately acquired and the missing signal is subsequently recovered through computational reconstruction. We transfer this acquisition–reconstruction principle to generative inference: expensive DiT evaluations are treated as explicit observations of the evolving generative state, and only a structured subset needs to be acquired if the complementary information can be reliably reconstructed. Fig. 2 provides three empirical motivations. First, intermediate generative states exhibit substantial cross-step and spatiotemporal redundancy. As shown in Fig. 2 (a), the change between adjacent clean-latent predictions rapidly decreases along the sampling trajectory, while spatially and temporally neighboring tokens remain far more similar than randomly paired tokens. This strong predictability motivates reconstructing

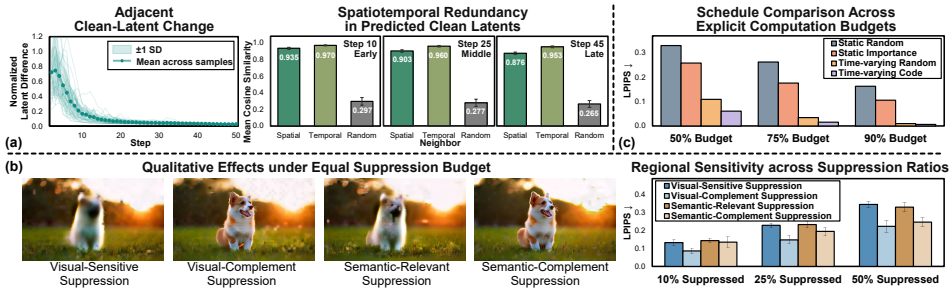

Figure 2: **Empirical motivation for coded computation.** (a) Adjacent clean-latent changes decrease along denoising, while spatially and temporally neighboring tokens remain substantially more similar than random pairs, revealing strong cross-step and spatiotemporal redundancy. (b) Under equal suppression budgets, masking visually sensitive or semantically relevant regions causes larger degradation than masking their complementary regions, demonstrating non-uniform computational value. (c) Static random, static importance, time-varying random, and time-varying coding produce markedly different errors under the same explicit-computation budgets, showing the importance of computation scheduling

generative information without explicitly recomputing every token at every step. Second, the value of explicit computation is highly content dependent. Fig. 2 (b) shows that, under the same suppression budget, masking visually sensitive or semantically relevant regions causes substantially larger degradation than masking their complementary regions. This motivates computation allocation based jointly on bottom-up visual cues and top-down semantic relevance. Third, how computation is distributed along the trajectory is itself important. Under the same explicit-computation budget, Fig. 2 (c) compares static random, static importance, time-varying random, and our time-varying coding schedules. The markedly different generation errors show that identifying important content alone is insufficient; computation must also be scheduled adaptively over diffusion steps.

Coded computation. We therefore formulate efficient video-diffusion inference as a joint problem of computational encoding and generative reconstruction, as illustrated in Fig. 3. At the entrance of a coded segment, a perceptual-semantic encoder $\mathcal{E}_\phi$ analyzes the current content and text condition:

$$(\mathbf{V}_x, \mathbf{S}_x, \mathbf{I}_x) = \mathcal{E}_\phi(\bar{z}_x, c), \tag{1}$$

where $\mathbf{V}_x$ denotes bottom-up visual cues, $\mathbf{S}_x$ denotes top-down semantic relevance, and $\mathbf{I}_x$ denotes their fused perceptual-semantic importance map. These signals characterize the content-dependent value of explicit computation, while a coded computation module estimates how reliably the omitted information can be reconstructed. Let $\mathbf{M}_t \in \{0,1\}^N$ denote the execution mask at step t, where $M_t(i) = 1$ indicates an explicit DiT evaluation for token i. The sequence

$$\mathcal{C}_{x\to y} = \big(\mathbf{M}_t\big)_{t\in\mathcal{T}_{x\to y}}, \tag{2}$$

defines the time-varying compute code as the ordered sequence of execution masks along the coded segment. As shown in Fig. 3, its realization jointly considers perceptual-semantic value, reconstruction reliability, and previously acquired observations. Selected tokens receive fresh DiT computation, whereas the complementary information is recovered by the code-aware generative decoder. The coded prediction is subsequently diffusion-state aligned and composed with the explicitly observed trajectory to form the complete exit state z_y. This formulation separates two complementary questions: what information is worth explicitly computing and what information can instead be reconstructed. Sec. 3.2 addresses the former, while Sec. 3.3 realizes their joint, time-varying computation and reconstruction.

3.2 Perceptual-Semantic Encoder

The perceptual-semantic encoder provides the content-dependent signals used to construct the time-varying compute code. As suggested by Fig. 2 (b), the value of explicit computation is highly content dependent, yet cannot be described by a single notion of importance. A visually sensitive region may require accurate computation even when it is not central to the prompt, whereas a small or visually simple object may still be critical to the intended generation. We therefore characterize

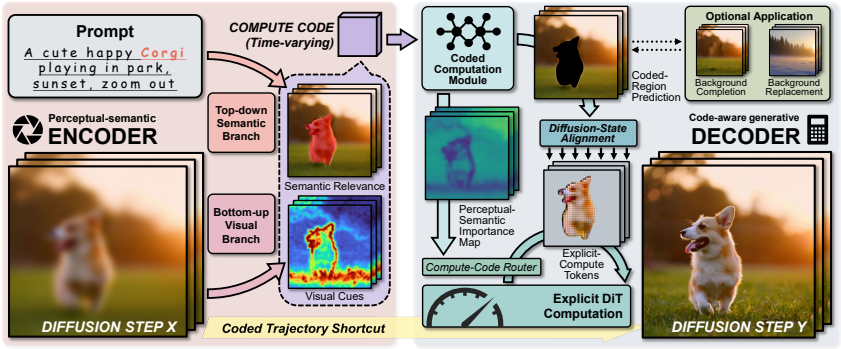

Figure 3: **Overview of the proposed coded-computation framework.** A perceptual-semantic encoder guides time-varying compute coding, while a code-aware generative decoder combines selective DiT computation with generative reconstruction to recover the complete video representation.

computational value using two complementary signals: bottom-up visual cues, which describe the visual structure and approximation sensitivity of the evolving content, and top-down semantic relevance, which reflects its importance to the generation intent.

Perceptual-semantic cues. The bottom-up visual branch B_{ϕ_v}, parameterized by ϕ_v, extracts visual cues $\mathbf{V}_x = B_{\phi_v}(\bar{z}_x)$ from the entrance clean estimate. The representation preserves the spatiotemporal layout of the latent video and captures appearance, texture, motion, local structure, and inter-frame variation. Besides providing content information to the subsequent coded computation module, an internal lightweight readout $g_{\text{vis}}(\mathbf{V}_x)$ measures the local sensitivity of these cues to computational approximation. Visual sensitivity alone, however, does not indicate whether content is important to the generation intent. We therefore construct a complementary top-down semantic signal. The top-down semantic branch uses $[\bar{z}_x, \mathbf{V}_x]$ to estimate a subject-geometry map $\mathbf{G}_x$, while token-to-text responses already available from DiT evaluations around the segment entrance are aggregated into a text-relevance map $\mathbf{T}_x$. After alignment to the DiT token grid, the two signals are fused in logit space:

$$\mathbf{S}_x = \text{sigmoid}\left[\lambda_{\text{geo}} \,\text{logit}(\mathbf{G}_x) + \lambda_{\text{text}} \,\text{logit}(\mathbf{T}_x)\right], \tag{3}$$

where λ_{geo} and λ_{text} control their relative contributions. The resulting $\mathbf{S}_x \in [0,1]^N$ is the semantic relevance map, emphasizing content that is both spatially associated with the generated subject and relevant to the text condition.

Time-varying compute coding. The two cues capture complementary notions of computational value: $g_{\text{vis}}(\mathbf{V}_x)$ measures where approximation is visually costly, whereas $\mathbf{S}_x$ measures where content matters to the intended generation. We fuse them into the perceptual-semantic importance map

$$\mathbf{I}_x = \Phi\left(g_{\text{vis}}(\mathbf{V}_x), \mathbf{S}_x\right) \in \mathbb{R}_+^N, \tag{4}$$

where Φ denotes the fixed weighted fusion operator. The resulting $\mathbf{I}_x$ encodes the content-dependent value of explicit computation over the spatiotemporal token grid. Importantly, we do not treat $\mathbf{I}_x$ as a static pruning mask. As observed in Fig. 2 (c), preserving important content alone is insufficient; how explicit computation is distributed along the diffusion trajectory is also critical. We therefore use $\mathbf{I}_x$ to condition the time-varying compute code $\mathcal{C}_{x\to y}$, whose realization is described in Sec. 3.3.

3.3 Code-Aware Generative Decoder

The perceptual-semantic importance map $\mathbf{I}_x$ specifies where explicit computation is valuable, but does not indicate whether the corresponding generative information actually needs to be recomputed. The code-aware generative decoder therefore complements computational value with reconstruction reliability, and uses the two to instantiate the time-varying compute code. It then combines sparse explicit DiT observations with a dense coded prediction to recover the complete generative state.

Coded computation module. Given the entrance clean-content estimate $\bar{z}_x$ and visual cues $\mathbf{V}_x$, the coded computation module G_ψ produces a dense clean-state prediction R over the complete spatiotemporal grid together with a token-wise reconstruction risk $\boldsymbol{\rho}_x$:

$$(R, \boldsymbol{\rho}_x) = G_\psi(\bar{z}_x, \mathbf{V}_x). \tag{5}$$

Here, R supplies the generative content used when explicit DiT computation is omitted, while $\boldsymbol{\rho}_x$ measures the risk of relying on this prediction at each token. Together with the perceptual-semantic importance $\mathbf{I}_x$, the two characterize what is valuable to compute and what is difficult to reconstruct.

Diffusion-state alignment. Sparse explicit computation must remain coupled with the complete video representation as the diffusion state evolves. We therefore construct layer-wise key-value references from both the dense prediction R and the segment-entrance observation. Let $\mathcal{B}_R$ denote the reference collection obtained from a DiT forward pass on R, and let $\mathcal{B}_x$ denote the corresponding references cached at the segment entrance. A noise-conditioned alignment operator $\mathcal{A}$ maps these references to the current diffusion state:

$$\widetilde{\mathcal{B}}_t = \mathcal{A}\left(\mathcal{B}_R, \mathcal{B}_x; x, t\right), \tag{6}$$

where $\widetilde{\mathcal{B}}_t$ provides a step-aligned contextual reference for sparse DiT computation. Previously acquired observation-derived corrections supplement this reference at locations supported by the observation history, while freshly evaluated tokens contribute their current keys and values. The resulting dense spatiotemporal context allows explicit-compute tokens to remain coupled with the full video representation throughout the coded segment.

Explicit DiT computation. We maintain an observation state $\mathcal{H}_t = \{\mathcal{H}_t(i)\}_{i=1}^{N}$, storing each token's explicit DiT observations and their acquisition steps before t. The compute-code router combines this history with perceptual-semantic importance and reconstruction risk:

$$\mathbf{M}_t = \Gamma\left(\mathbf{I}_x, \boldsymbol{\rho}_x, \mathcal{H}_t; \tau, \beta\right), \tag{7}$$

where $\tau > 0$ sets the reconstruction-error tolerance for identifying tokens requiring explicit tracking, and $\beta > 0$ controls their refreshes based on importance, risk, and observation history. For a fixed observation state, increasing β raises the refresh requirement. The ordered masks $(\mathbf{M}_t)_{t\in\mathcal{T}_{x\to y}}$ instantiate the time-varying compute code $\mathcal{C}_{x\to y}$ in Eq. (2).

Tokens with $M_t(i) = 1$ receive a fresh evaluation from the frozen DiT F_θ, yielding a clean-state prediction $\widehat{z}_t^{\mathrm{DiT}}(i)$ and its correction relative to R:

$$\boldsymbol{\delta}_t(i) = \widehat{z}_t^{\mathrm{DiT}}(i) - R(i). \tag{8}$$

Only fresh DiT observations update $\mathcal{H}_t$; skipped locations are reconstructed from R and any available observation history. The sampler combines these contributions into a complete latent state throughout the coded segment, ultimately producing the exit state z_y.

Optional background completion and replacement. The clean-state prediction R also provides a convenient interface for optional background editing. External or completed background content can be encoded into the latent space and used to supplement or replace designated background regions of R before diffusion-state alignment. The modified prediction then follows the same alignment and code-aware reconstruction pipeline, with computation demand reassessed around the affected regions and their boundaries.

3.4 Training and Efficient Inference

Training. We freeze the pretrained DiT F_θ and collect dense generation trajectories as supervision. The auxiliary visual and subject predictors in Sec. 3.2 are first trained using their corresponding feature and segmentation supervision and are kept fixed thereafter. We then train the coded computation module G_ψ from pairs of entrance estimates $\bar{z}_x$ and teacher clean-state targets $R^\star$ obtained from dense inference. The training objective is

$$\mathcal{L} = \lambda_{\mathrm{rec}}\mathcal{L}_{\mathrm{rec}} + \lambda_{\mathrm{tem}}\mathcal{L}_{\mathrm{tem}} + \lambda_{\mathrm{per}}\mathcal{L}_{\mathrm{per}} + \lambda_{\mathrm{qua}}\mathcal{L}_{\mathrm{qua}}, \tag{9}$$

where $\mathcal{L}_{\mathrm{rec}}$ supervises latent and decoded-image reconstruction, $\mathcal{L}_{\mathrm{tem}}$ preserves temporal consistency, $\mathcal{L}_{\mathrm{per}}$ encourages perceptual and structural fidelity, and $\mathcal{L}_{\mathrm{qua}}$ supervises the reconstruction-quality estimates used to form $\boldsymbol{\rho}_x$. After training, these quality estimates are calibrated on a held-out set before being consumed by the compute-code router.

Efficient inference. At inference time, the perceptual-semantic encoder and coded computation module are evaluated at the entrance of each coded segment to obtain $\mathbf{I}_x$, R, and $\boldsymbol{\rho}_x$. The compute-code router then updates $\mathbf{M}_t$ as new explicit observations are incorporated into $\mathcal{H}_t$, and only selected tokens are gathered for expensive DiT evaluation. The resulting observations are combined with the aligned dense prediction. Technical details and training configurations are provided in Appendices A–D.

Table 1: **Quantitative comparison of video diffusion acceleration methods.** Results are reported on Wan2.1-1.3B and LTX-Video-2B. Bold indicates the best result among acceleration methods for each model.

Model	Method	Latency (s) ↓	Speedup ↑	LPIPS ↓	SSIM ↑	PSNR ↑	VBench (%) ↑
Wan2.1-1.3B	Native	99.31	1.00×	–	–	–	79.68
	TeaCache	53.35	1.86×	0.028	0.936	31.33	**79.84**
	ProfilingDiT	46.53	2.13×	0.055	0.897	28.31	79.35
	SenCache	38.90	2.55×	0.027	0.937	31.86	79.67
	NaviCache	40.81	2.43×	0.025	0.930	31.49	79.66
	Ours-Low	69.87	1.42×	**0.006**	**0.973**	**38.68**	79.64
	Ours-Mid	41.64	2.38×	0.024	0.942	32.75	79.53
	Ours-High	**37.55**	**2.64**×	0.033	0.928	31.27	79.47
LTX-Video-2B	Native	50.20	1.00×	–	–	–	77.88
	TeaCache	32.88	1.53×	0.056	0.919	28.78	77.58
	ProfilingDiT	26.95	1.86×	0.098	0.877	26.56	77.08
	SenCache	25.90	1.94×	0.032	0.928	28.48	77.51
	NaviCache	41.88	1.20×	0.006	0.985	39.54	**77.93**
	Ours-Low	33.64	1.49×	**0.003**	**0.990**	**42.35**	77.92
	Ours-Mid	23.31	2.15×	0.019	0.967	34.76	77.83
	Ours-High	**20.28**	**2.48**×	0.029	0.954	32.37	77.80

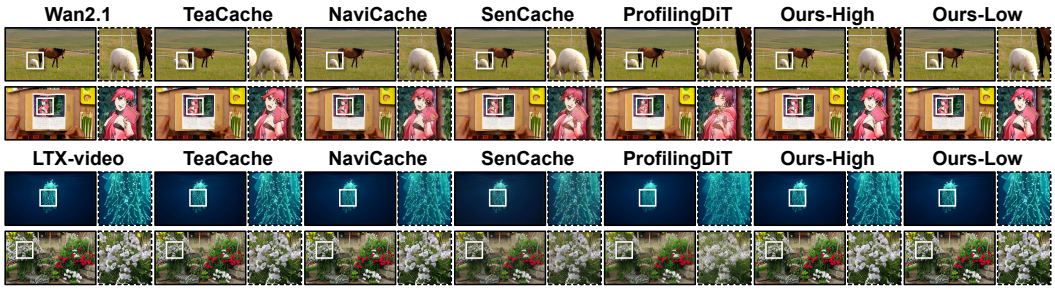

Figure 4: **Qualitative comparison with existing acceleration methods.** The upper and lower groups show results on Wan2.1 and LTX-Video, respectively. Representative frames and enlarged crops highlight differences in subject structure and fine details relative to native generation.

4 Experiments

4.1 Experimental Setup

Base models and baselines. We evaluate our method on Wan2.1-1.3B (Team Wan et al., 2025) and LTX-Video-2B (HaCohen et al., 2024). We compare with native inference and four acceleration baselines that aim to preserve the native generated content: TeaCache (Liu et al., 2025a), ProfilingDiT (Ma et al., 2025b), SenCache (Haghighi & Alahi, 2026), and NaviCache (Lv et al., 2026). These methods retain the backbone's native weights and generation conditions, reducing inference cost through computation reuse or selective execution. We follow official baseline implementations and configurations as closely as possible.

Evaluation data and metrics. We use the full VBench prompt suite (Huang et al., 2024), with matched prompts and random seeds across methods. Wan encoder and decoder ablations use the same prompts and seeds as the main comparison. LPIPS (Zhang et al., 2018), SSIM (Wang et al., 2004), and PSNR measure fidelity to native outputs, while VBench evaluates generation quality. We report mean wall-clock latency per video and compute speedup as the ratio of native to accelerated latency under matched hardware and generation settings.

Implementation details. Main comparisons are conducted on an NVIDIA GH200 platform (96 GB GPU memory), and ablations on an NVIDIA A800 GPU (80 GB). Both backbones use 50 denoising steps. Wan generates 81 frames at 832×480 resolution with a classifier-free guidance (CFG) (Ho & Salimans, 2022) scale of 5.0, while LTX generates 121 frames at 1216×704 with a CFG scale of 3.0. Our model uses the first 10 steps for dense warm-up and applies compute-code routing to the remaining 40 steps. We evaluate three acceleration settings: Low, Mid, and High, corresponding to

Table 2: **Ablation of the perceptual-semantic encoder.** Visual and Semantic use the encoder's corresponding cues for random sampling or history coverage. All variants share the candidate set, decoder, and per-step update budget. Ours retains the complete adaptive allocation rule. Fidelity is measured against native outputs.

Configuration		LPIPS ↓	SSIM ↑	PSNR ↑
Random		0.081	0.869	26.68
ROI random		0.276	0.670	18.81
History coverage		0.065	0.882	28.09
Visual	random	0.241	0.724	21.50
	coverage	0.052	0.901	29.23
Semantic	random	0.239	0.726	21.24
	coverage	0.052	0.901	29.16
Ours		**0.033**	**0.928**	**31.27**

Table 3: **Ablation of the code-aware generative decoder.** State and Context toggle learned reconstruction and learned reference content, using fixed history extrapolation as the replacement. Alignment toggles diffusion-state alignment of references. All variants share execution masks, with fidelity measured against native outputs.

Mode	State	Context	Alignment	LPIPS ↓	SSIM ↑	PSNR ↑
A0	×	×	×	0.430	0.625	19.57
A1	✓	×	×	0.355	0.685	20.62
A2	×	✓	×	0.214	0.765	22.88
A3	✓	✓	×	0.109	0.828	25.11
A4	×	×	✓	0.200	0.771	22.54
A5	✓	×	✓	0.105	0.832	24.68
A6	×	✓	✓	0.146	0.864	26.52
Ours	✓	✓	✓	**0.033**	**0.928**	**31.27**

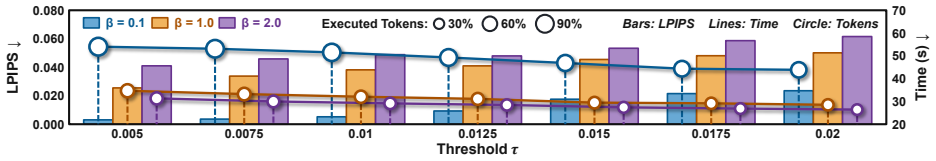

Figure 5: **Sensitivity to compute-code routing parameters.** We evaluate a single prompt while varying τ and β to examine the trade-off between computation and reconstruction fidelity. At $\tau = 0.01$, the Low, Mid, and High acceleration settings of our model correspond to $\beta = 0.1$, 1.0, and 2.0, respectively.

$\beta = 0.1$, 1.0, and 2.0, respectively. Additional experimental configurations, results, and analyses are provided in Appendices E–K.

4.2 Main Results

Quantitative comparison. Table 1 shows that Low achieves the best LPIPS, SSIM, and PSNR on both backbones. On LTX, Low improves PSNR over TeaCache by 13.57 dB at comparable latency. Mid provides stronger acceleration with competitive fidelity: on Wan, it improves all three fidelity metrics over NaviCache at comparable latency, including a 1.26 dB PSNR gain; on LTX, it reduces LPIPS and latency by approximately 41% and 10.0% relative to SenCache. Mid also achieves better fidelity and lower latency than ProfilingDiT on both backbones, supporting the effectiveness of combining selective computation with learned reconstruction. High achieves the largest speedups in the comparison, 2.64× on Wan and 2.48× on LTX, with VBench decreases of 0.21 and 0.08 percentage points relative to native inference, respectively.

Visualization. Fig. 4 shows that our results most closely resemble native outputs on both Wan and LTX, preserving semantic content and fine-grained appearance. On Wan, our method maintains the native object count and character details; on LTX, it better preserves tentacle contours, petal boundaries, and local textures.

4.3 Ablation Studies and Analysis

Perceptual-semantic encoder. Table 2 ablates how encoder cues guide computation, fixing the candidate set, decoder, and per-step update budget. Under the same coverage rule, either visual or semantic guidance reduces LPIPS from 0.0653 to 0.0521 (20.2%), showing that each cue improves the selection of computation locations. The full rule integrates continuous perceptual-semantic cues with reconstruction risk and observation feedback, reaching 0.0330 LPIPS, a further 36.7% reduction. These results support the encoder's content guidance and its integration with evolving reconstruction demand.

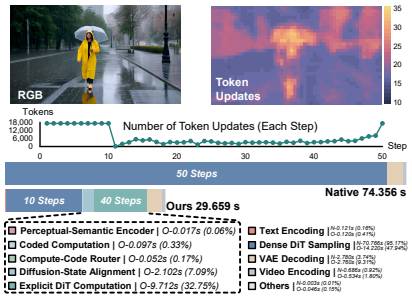

Figure 6: **Token-update distribution and runtime breakdown.** High-setting profile for a single prompt. Top: a generated frame, cumulative spatial updates, and updates per denoising step. Bottom: component runtimes for Native (N) and Ours (O).



Figure 7: **Background completion and replacement through coded reconstruction.** Blue: original accelerated videos and their scene descriptions. Green: completed backgrounds and replacement results with their edited scene descriptions.

Code-aware generative decoder. Table 3 ablates the coded computation module and diffusion-state alignment in Sec. 3.3 under identical execution masks. Replacing the module's learned state-reconstruction prediction with history extrapolation raises LPIPS from 0.0330 to 0.1456 (A6). Using history extrapolation to construct alignment references (A5) also degrades fidelity, showing that the learned prediction supports both state recovery and reference construction. Disabling diffusion-state alignment (A3) likewise degrades fidelity, indicating that references must match the current noise level. These results support their complementary roles in recovering uncomputed content and coupling explicit DiT computation with the complete video representation.

Sensitivity to routing parameters. Fig. 5 varies the reconstruction-error tolerance τ and refresh threshold β. Increasing either generally reduces token execution and latency at the cost of higher LPIPS. For larger τ, latency gains diminish while fidelity continues to degrade.

Adaptive computation and efficiency. Fig. 6 shows that, after ten dense warm-up steps, our method refreshes only 22.4% of tokens per step on average, concentrating updates on the subject and detailed background regions. This selective computation substantially reduces runtime during the remaining forty steps, yielding a 60.1% reduction in end-to-end latency. The additional overhead is only 2.27 s: 2.10 s for reference construction and alignment, and 0.17 s for the encoder, coded computation, and router combined.

4.4 Optional Applications

Background completion and replacement. Fig. 7 demonstrates background editing through the dense clean-state prediction R. Inpainted or generated backgrounds are encoded and inserted into the background regions of R before diffusion-state alignment and code-aware reconstruction. Foreground compositing helps preserve the source subject while allowing changes to the surrounding scenery and illumination.

5 Conclusion

We presented *coded computation*, a computational-imaging-inspired framework combining selective DiT computation with code-aware reconstruction for efficient video diffusion. Guided by perceptual-semantic importance, reconstruction reliability, and observation history, our method achieves up to $2.64\times$ and $2.48\times$ acceleration on Wan2.1-1.3B and LTX-Video-2B, respectively, with near-native generation quality. The reconstructed clean-state representation also supports background completion and replacement.

Limitations. Our current evaluation focuses on representative open video diffusion backbones and primarily studies inference acceleration under text-to-video generation. Extending coded computation to broader generative architectures, longer-form video generation, and other conditional generation settings remains an interesting direction for future work.

AI USE STATEMENT

Generative AI tools were used to aid or polish writing, including improving clarity and presentation, checking numerical consistency, and assisting with LaTeX formatting. The authors reviewed all AI-assisted content and take full responsibility for the scientific claims, reported results, and final content of this paper.

REPRODUCIBILITY STATEMENT

Section 4.1 describes the evaluation settings, metrics, and prompt and random-seed protocol. Appendices A–C document the model components, inference procedures, and training settings. Baseline configurations are provided in Appendix E.1, while Appendices G–I detail the encoder and decoder ablations and routing sensitivity experiments.

REFERENCES

Ayush Bhandari, Achuta Kadambi, and Ramesh Raskar. *Computational Imaging*. MIT Press, 2022. ISBN 9780262046473. URL https://mitpress.mit.edu/9780262046473/computational-imaging/.

Jiazi Bu, Pengyang Ling, Yujie Zhou, Yibin Wang, Yuhang Zang, Dahua Lin, and Jiaqi Wang. DiCache: Let diffusion model determine its own cache. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/78288ef33b18a351c3cd679dc9a15c8d-Abstract-Conference.html.

Guantao Chen, Shikang Zheng, Yuqi Lin, and Linfeng Zhang. Forecast the principal, stabilize the residual: Subspace-aware feature caching for diffusion transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23632–23641, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Chen_Forecast_the_Principal_Stabilize_the_Residual_Subspace-Aware_Feature_Caching_for_CVPR_2026_paper.html.

Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. In *European Conference on Computer Vision*, pp. 17–33, 2022. doi: 10.1007/978-3-031-20071-7_2. URL https://link.springer.com/chapter/10.1007/978-3-031-20071-7_2.

Ernie Chu and Vishal M. Patel. Not all tokens need 40 steps: Heterogeneous step allocation in diffusion transformers for efficient video generation. arXiv preprint arXiv:2605.06892, 2026. URL https://arxiv.org/abs/2605.06892.

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.

Kanchana Vaishnavi Gandikota, Michael Moeller, Andreas Kolb, Bhaskar Choubey, and Paramanand Chandramouli. A bit is all you need! Efficient video capture via single bit imaging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 34016–34027, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Gandikota_A_Bit_is_All_You_Need_Efficient_Video_Capture_via_CVPR_2026_paper.html.

Huanlin Gao, Ping Chen, Fuyuan Shi, Chao Tan, Zhaoxiang Liu, Fang Zhao, Kai Wang, and Shiguo Lian. LeMiCa: Lexicographic minimax path caching for efficient diffusion-based video generation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 132388–132415, 2025. doi: 10.52202/085713-4408. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/bfa0d4d182de7e7939f67940fe524cce-Abstract-Conference.html.

Xingtong Ge, Yi Zhang, Yushi Huang, Dailan He, Xiahong Wang, Bingqi Ma, Guanglu Song, Yu Liu, and Jun Zhang. Salt: Self-consistent distribution matching with cache-aware training for fast video generation. arXiv preprint arXiv:2604.03118, 2026. URL https://arxiv.org/abs/2604.03118.

Yuchao Gu, Guian Fang, Yuxin Jiang, Weijia Mao, Song Han, Han Cai, and Mike Zheng Shou. AnyFlow: Any-step video diffusion model with on-policy flow map distillation. arXiv preprint arXiv:2605.13724, 2026. URL https://arxiv.org/abs/2605.13724.

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, Poriya Panet, Sapir Weissbuch, Victor Kulikov, Yaki Bitterman, Zeev Melumian, and Ofir Bibi. LTX-Video: Realtime video latent diffusion. arXiv preprint arXiv:2501.00103, 2024. URL https://arxiv.org/abs/2501.00103.

Yasaman Haghighi and Alexandre Alahi. SenCache: Accelerating diffusion model inference via sensitivity-aware caching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14295–14304, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Haghighi_SenCache_Accelerating_Diffusion_Model_Inference_via_Sensitivity-Aware_Caching_CVPR_2026_paper.html.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. arXiv preprint arXiv:2207.12598, 2022. URL https://arxiv.org/abs/2207.12598.

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, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Huang_VBench_Comprehensive_Benchmark_Suite_for_Video_Generative_Models_CVPR_2024_paper.html.

Junha Hyung, Kinam Kim, Susung Hong, Min-Jung Kim, and Jaegul Choo. Spatiotemporal skip guidance for enhanced video diffusion sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11006–11015, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Hyung_Spatiotemporal_Skip_Guidance_for_Enhanced_Video_Diffusion_Sampling_CVPR_2025_paper.html.

Zeyinzi Jiang, Zhen Han, Chaojie Mao, Jingfeng Zhang, Yulin Pan, and Yu Liu. VACE: All-in-one video creation and editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17191–17202, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Jiang_VACE_All-in-One_Video_Creation_and_Editing_ICCV_2025_paper.html.

Kumara Kahatapitiya, Haozhe Liu, Sen He, Ding Liu, Menglin Jia, Chenyang Zhang, Michael S. Ryoo, and Tian Xie. Adaptive caching for faster video generation with diffusion transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15240–15252, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Kahatapitiya_Adaptive_Caching_for_Faster_Video_Generation_with_Diffusion_Transformers_ICCV_2025_paper.html.

Jeremy Klotz and Shree K. Nayar. Minimalist vision with freeform pixels. In *European Conference on Computer Vision*, pp. 329–346, 2024. doi: 10.1007/978-3-031-73039-9_19. URL https://link.springer.com/chapter/10.1007/978-3-031-73039-9_19.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, Kathrina Wu, Qin Lin, Junkun Yuan, Yanxin Long, Aladdin Wang, Andong Wang, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Hongmei Wang, Jacob Song, Jiawang Bai, Jianbing Wu, Jinbao Xue, Joey Wang, Kai Wang, Mengyang Liu, Pengyu Li, Shuai Li, Weiyan Wang, Wenqing Yu, Xinchi Deng, Yang Li, Yi Chen, Yutao Cui, Yuanbo Peng, Zhentao Yu, Zhiyu He, Zhiyong Xu, Zixiang Zhou, Zunnan Xu, Yangyu Tao, Qinglin Lu, Songtao

Liu, Dax Zhou, Hongfa Wang, Yong Yang, Di Wang, Yuhong Liu, Jie Jiang, and Caesar Zhong. HunyuanVideo: A systematic framework for large video generative models. arXiv preprint arXiv:2412.03603, 2024. URL https://arxiv.org/abs/2412.03603.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012. URL https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

Anat Levin, Rob Fergus, Frédo Durand, and William T. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM Transactions on Graphics*, 26(3):70, 2007. doi: 10.1145/1276377.1276464. URL https://groups.csail.mit.edu/graphics/CodedAperture/.

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, 2026. URL https://arxiv.org/abs/2605.17837.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7353–7363, 2025a. URL https://openaccess.thecvf.com/content/CVPR2025/html/Liu_Timestep_Embedding_Tells_Its_Time_to_Cache_for_Video_Diffusion_CVPR_2025_paper.html.

Haosong Liu, Yuge Cheng, Wenxuan Miao, Zihan Liu, Aiyue Chen, Jing Lin, Yiwu Yao, Chen Chen, Jingwen Leng, Minyi Guo, and Yu Feng. Astraea: A token-wise acceleration framework for video diffusion transformers. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/921ac785fa9edc73cacaf2664f43d234-Abstract-Conference.html.

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*, pp. 15853–15863, 2025b. URL https://openaccess.thecvf.com/content/ICCV2025/html/Liu_From_Reusing_to_Forecasting_Accelerating_Diffusion_Models_with_TaylorSeers_ICCV_2025_paper.html.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Yihong Luo, Tianyang Hu, Jiacheng Sun, Yujun Cai, and Jing Tang. Learning few-step diffusion models by trajectory distribution matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17719–17728, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Luo_Learning_Few-Step_Diffusion_Models_by_Trajectory_Distribution_Matching_ICCV_2025_paper.html.

Zhengyao Lv, Chenyang Si, Tianlin Pan, Zhaoxi Chen, Kwan-Yee K. Wong, Yu Qiao, and Ziwei Liu. Dual-expert consistency model for efficient and high-quality video generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14983–14993, 2025a. URL https://openaccess.thecvf.com/content/ICCV2025/html/Lv_Dual-Expert_Consistency_Model_for_Efficient_and_High-Quality_Video_Generation_ICCV_2025_paper.html.

Zhengyao Lv, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K. Wong. FasterCache: Training-free video diffusion model acceleration with high quality. In *International Conference on Learning Representations*, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/518046d86bbc41a0707727c38301ad8e-Abstract-Conference.html.

Zheqi Lv, Zhibo Zhu, Jinke Wang, Qi Tian, Shengyu Zhang, Zhengyu Chen, Chengxi Zang, Zhou Zhao, and Fei Wu. NaviCache: Test-time self-calibration caching for video generation. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026. URL https://arxiv.org/abs/2606.26795.

Xin Ma, Yaohui Wang, Xinyuan Chen, Gengyun Jia, Ziwei Liu, Yuan-Fang Li, Cunjian Chen, and Yu Qiao. Latte: Latent diffusion transformer for video generation. *Transactions on Machine Learning Research*, 2025a. URL https://openreview.net/forum?id=ntGPYNUF3t.

Xuran Ma, Yexin Liu, Yaofu Liu, Xianfeng Wu, Mingzhe Zheng, Zihao Wang, Ser-Nam Lim, and Harry Yang. Model reveals what to cache: Profiling-based feature reuse for video diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17150–17159, 2025b. URL https://openaccess.thecvf.com/content/ICCV2025/html/Ma_Model_Reveals_What_to_Cache_Profiling-Based_Feature_Reuse_for_Video_ICCV_2025_paper.html.

Zehong Ma, Longhui Wei, Feng Wang, Shiliang Zhang, and Qi Tian. MagCache: Fast video generation with magnitude-aware cache. In *Advances in Neural Information Processing Systems*, volume 38, pp. 34348–34380, 2025c. doi: 10.52202/085713-1153. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/311207bb626e36a8f1d3eb92aa67af22-Abstract-Conference.html.

Julien N. P. Martel, Lorenz K. Müller, Stephen J. Carey, Piotr Dudek, and Gordon Wetzstein. Neural sensors: Learning pixel exposures for HDR imaging and video compressive sensing with programmable sensors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42 (7):1642–1653, 2020. doi: 10.1109/TPAMI.2020.2986944. URL https://ieeexplore.ieee.org/document/9064896.

Christopher A. Metzler, Hayato Ikoma, Yifan Peng, and Gordon Wetzstein. Deep optics for single-shot high-dynamic-range imaging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1375–1385, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Metzler_Deep_Optics_for_Single-Shot_High-Dynamic-Range_Imaging_CVPR_2020_paper.html.

Weili Nie, Julius Berner, Nanye Ma, Chao Liu, Saining Xie, and Arash Vahdat. Transition matching distillation for fast video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4645–4655, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Nie_Transition_Matching_Distillation_for_Fast_Video_Generation_CVPR_2026_paper.html.

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

Ramesh Raskar, Amit Agrawal, and Jack Tumblin. Coded exposure photography: Motion deblurring using fluttered shutter. *ACM Transactions on Graphics*, 25(3):795–804, 2006. doi: 10.1145/1141911.1141957. URL https://www.merl.com/publications/TR2006-124.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/45c1f6a8cbf2da59ebf2c802b4f742cd-Abstract-Conference.html.

Neta Shaul, Chao Liu, Arash Vahdat, and Julius Berner. Parallel decoding distillation for fast image and video generation. arXiv preprint arXiv:2607.26004, 2026. URL https://arxiv.org/abs/2607.26004.

Zhirong Shen, Rui Huang, Jiacheng Liu, Chang Zou, Peiliang Cai, Shikang Zheng, Zhengyi Shi, Liang Feng, and Linfeng Zhang. Beyond fixed formulas: Data-driven linear predictor for efficient diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30792–30801, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Shen_Beyond_Fixed_Formulas_Data-Driven_Linear_Predictor_for_Efficient_Diffusion_Models_CVPR_2026_paper.html.

Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1874–1883, 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/html/Shi_Real-Time_Single_Image_CVPR_2016_paper.html.

Jibin Song, Mingi Kwon, Jaeseok Jeong, and Youngjung Uh. FlowBlending: Stage-aware multi-model sampling for fast and high-fidelity video generation. arXiv preprint arXiv:2512.24724, 2025. URL https://arxiv.org/abs/2512.24724.

Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with Fourier convolutions. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2149–2159, 2022. URL https://openaccess.thecvf.com/content/WACV2022/html/Suvorov_Resolution-Robust_Large_Mask_Inpainting_With_Fourier_Convolutions_WACV_2022_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.

Edwin Vargas, Julien N. P. Martel, Gordon Wetzstein, and Henry Arguello. Time-multiplexed coded aperture imaging: Learned coded aperture and pixel exposures for compressive imaging systems. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2692–2702, 2021. URL https://openaccess.thecvf.com/content/ICCV2021/html/Vargas_Time-Multiplexed_Coded_Aperture_Imaging_Learned_Coded_Aperture_and_Pixel_Exposures_ICCV_2021_paper.html.

Ao Wang, Lihao Liu, Hui Chen, Zijia Lin, Jungong Han, and Guiguang Ding. YOLOE: Real-time seeing anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 24591–24602, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Wang_YOLOE_Real-Time_Seeing_Anything_ICCV_2025_paper.html.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. doi: 10.1109/TIP.2003.819861. URL https://eceweb.uwaterloo.ca/~z70wang/publications/ssim.html.

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, Jianfei Chen, Ion Stoica, Kurt Keutzer, and Song Han. Sparse VideoGen: Accelerating video diffusion transformers with spatial-temporal sparsity. In *Proceedings of the 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.

Howard Xiao, Jan Ackermann, Boyang Deng, and Gordon Wetzstein. Policy-based foveated imaging and perception. arXiv preprint arXiv:2606.02565, 2026. URL https://arxiv.org/abs/2606.02565.

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. 96965–96991, 2025a. doi: 10.52202/085713-3243. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/8bd5148caced2d73cea7b6961a874a49-Abstract-Conference.html.

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. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/ce31378e9f41d8907e97dab172b6c559-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*, 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.

Jintao Zhang, Xiaoming Xu, Jia Wei, Haofeng Huang, Pengle Zhang, Chendong Xiang, Jun Zhu, and Jianfei Chen. SageAttention2++: A more efficient implementation of SageAttention2. arXiv preprint arXiv:2505.21136, 2025a. URL https://arxiv.org/abs/2505.21136.

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

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_The_Unreasonable_Effectiveness_CVPR_2018_paper.html.

Wenhu Zhang, Kun Cheng, Changyuan Wang, Shiyao Li, Yuechen Zhang, Wenbo Li, Jiajun Zha, Jingyi Zhang, Kang Zhao, and Jiaya Jia. CoDMD: Copula-aware distribution matching distillation for fast video generation. arXiv preprint arXiv:2606.21982, 2026. URL https://arxiv.org/abs/2606.21982.

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*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a4a1ee071ce0fe63b83bce507c9dc4d7-Abstract-Conference.html.

Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with Pyramid Attention Broadcast. In *International Conference on Learning Representations*, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/092c2d45005ea2db40fc24c470663416-Abstract-Conference.html.

Yaping Zhao and Edmund Y. Lam. SASA: Saliency-aware self-adaptive snapshot compressive imaging. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2370–2374, 2024. doi: 10.1109/ICASSP48485.2024.10446309. URL https://arxiv.org/abs/2401.00875.

Kaiwen Zheng, Yuji Wang, Qianli Ma, Huayu Chen, Jintao Zhang, Yogesh Balaji, Jianfei Chen, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Large scale diffusion distillation via score-regularized continuous-time consistency. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0534abc9e6db91683d82186ef0d68202-Abstract-Conference.html.

Chu Zhou, Hang Zhao, Jin Han, Chang Xu, Chao Xu, Tiejun Huang, and Boxin Shi. UnModNet: Learning to unwrap a modulo image for high dynamic range imaging. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1559–1570, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1102a326d5f7c9e04fc3c89d0ede88c9-Abstract.html.

Chu Zhou, Yufei Han, Minggui Teng, Jin Han, Si Li, Chao Xu, and Boxin Shi. Polarization guided HDR reconstruction via pixel-wise depolarization. *IEEE Transactions on Image Processing*, 32: 1774–1787, 2023. doi: 10.1109/TIP.2023.3251023. URL https://ieeexplore.ieee.org/document/10061479/.

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/bbe024e0517fe12ac3a8d388b19ff9fe-Abstract-Conference.html.

Chang Zou, Changlin Li, Songtao Liu, Zhao Zhong, Kailin Huang, and Linfeng Zhang. DisCa: Accelerating video diffusion transformers with distillation-compatible learnable feature caching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4590–4601, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Zou_DisCa_Accelerating_Video_Diffusion_Transformers_with_Distillation-Compatible_Learnable_Feature_Caching_CVPR_2026_paper.html.

A PERCEPTUAL-SEMANTIC ENCODER

We detail the Wan implementation and the LTX adaptations of the encoder–decoder framework introduced in the main text.

A.1 VISUAL BRANCH

The visual branch takes the clean-state prediction $\bar{z}_x$ at the coded-segment entrance x. In the current configuration, the coded segment starts after the tenth DiT evaluation. The input latent has $C_z = 16$ channels for Wan and $C_z = 128$ for LTX, with DiT patches of $1 \times 2 \times 2$ and $1 \times 1 \times 1$, respectively.

Visual features. The branch produces a 46-channel feature map $\mathbf{F}_x$ and a single-channel detail map $\mathbf{H}_x$, forming the visual representation $\mathbf{V}_x = [\mathbf{F}_x, \mathbf{H}_x]$. It comprises a $C_z \to 48$ 3D convolution, four Temporal NAF blocks, channel-wise LayerNorm, and a $48 \to 47$ projection. A sigmoid applied to the last output channel gives $\mathbf{H}_x$. Each Temporal NAF block augments a NAFNet (Chen et al., 2022) block with a $3 \times 1 \times 1$ temporal depthwise convolution. Wan conditions this branch only on $\bar{z}_x$; LTX additionally injects the entrance noise level σ_x through a $1 \to 48 \to 48$ MLP.

Grid readout. Let $\mathcal{P}$ map the latent grid to the DiT token grid: it is $1 \times 2 \times 2$ average pooling for Wan and the identity for LTX. The detail readout is $\mathbf{h}_x = g_{\text{vis}}(\mathbf{V}_x) = \mathcal{P}(\mathbf{H}_x)$.

A.2 SEMANTIC BRANCH

Wan geometry prediction. The concatenated input $[\bar{z}_x, \mathbf{V}_x]$ has 63 channels and is projected to 32 channels. Four Temporal NAF blocks, LayerNorm, a single-channel projection, and a sigmoid produce the latent-grid geometry map $\mathbf{G}_x^{\text{lat}}$.

Wan text relevance. We set $\mathbf{G}_x = \mathcal{P}(\mathbf{G}_x^{\text{lat}})$ and collect conditional-branch text responses at step 10 from layers 5, 10, 15, and 20, using zero-based layer indices. Subtokens belonging to the same word are merged, function words are removed, and each phrase key is centered by subtracting the mean prompt key. Query–key responses are averaged over attention heads. Each phrase's spatial response is clipped and normalized using its 5th and 95th percentiles, then averaged across layers. Let i index DiT tokens and j index phrases, with response $A_x(i,j)$; $G_x(i)$ is the value of $\mathbf{G}_x$ at token i.

$$
\begin{aligned}
a_x(j) &= \frac{\sum_i A_x(i,j) G_x(i)}{\max(\sum_i G_x(i), 10^{-6})} - \frac{\sum_i A_x(i,j)(1 - G_x(i))}{\max(\sum_i (1 - G_x(i)), 10^{-6})}, \\
T_x(i) &= \sum_j \frac{[a_x(j)]_+^2}{\sum_k [a_x(k)]_+^2} A_x(i,j).
\end{aligned}
\tag{A1}
$$

If no phrase is available or the sum of positive weights is at most 10^{-12}, we set $T_x(i) = 0.5$. After clipping $\mathbf{G}_x$ and $\mathbf{T}_x$ to $[0.005, 0.995]$, we use the logit-space fusion in main-paper Eq. (3), with $\lambda_{\text{geo}} = 1.5$ and $\lambda_{\text{text}} = 0.5$. The binary subject map $\mathbf{b}_x$ is obtained from frame-wise 8-connected components of $\mathbf{S}_x$, using a low threshold of 0.4 and a high threshold of 0.6. Each retained component contains at least four tokens and at least one token reaching the high threshold.

LTX text relevance. We use conditional-branch responses from layers 6, 13, and 20, excluding padding and the first text token. After the same percentile normalization, responses are weighted by each text token's original spatial response variance and averaged across layers to obtain $\mathbf{T}_x$. This instantiates Eq. (3) with $\lambda_{\text{geo}} = 0$ and $\lambda_{\text{text}} = 1$, giving $\mathbf{S}_x = \mathbf{T}_x$. This text-only semantic branch requires neither a geometry head nor connected-component filtering. Missing valid responses are replaced by 0.5.

A.3 PERCEPTUAL-SEMANTIC IMPORTANCE

The two cues are fused on the token grid into $\mathbf{I}_x$. Here, $S_x(i)$, $h_x(i)$, and $I_x(i)$ denote the corresponding map values at token i:

$$
\begin{aligned}
\mathbf{I}_x &= \Phi(g_{\text{vis}}(\mathbf{V}_x), \mathbf{S}_x) = \mathbf{1} + 3\mathbf{S}_x + 0.5\mathbf{h}_x, \\
I_x(i) &= 1 + 3S_x(i) + 0.5h_x(i), \qquad \mathbf{h}_x = g_{\text{vis}}(\mathbf{V}_x).
\end{aligned}
\tag{A2}
$$

Equation (A2) gives the implementation of Φ in main-paper Eq. (4): a fixed weighted sum of bounded visual and semantic cues. The cue values lie in $[0, 1]$, while the fused importance weights lie in $[1, 4.5]$ and enter the routing demand directly. No further sum or min–max normalization is applied to $\mathbf{I}_x$.

B Code-Aware Generative Decoder

B.1 Dense Prediction and Reconstruction Network

Given the entrance clean estimate $\bar{z}_x$ and visual cues $\mathbf{V}_x$, the reconstruction module predicts a dense clean state R, an RGB-error map $\mathbf{e}_x$, and an auxiliary latent-risk map $\mathbf{u}_x$. As detailed in Sec. B.2, the predicted error is calibrated and combined with entrance-state changes, semantic and visual cues, and the tolerance τ to form the reconstruction risk $\boldsymbol{\rho}_x$ in main-paper Eq. (5). The router uses $\boldsymbol{\rho}_x$ and $\mathbf{u}_x$ together with perceptual-semantic importance and observation history to schedule explicit DiT evaluations.

Reconstruction backbone. The four encoder stages have widths $[64, 128, 256, 512]$ and block counts $[1, 1, 1, 8]$. The bottleneck has width 1024 and six blocks; each of the four decoder stages contains one block. Downsampling is spatial only; upsampling uses spatial PixelShuffle (Shi et al., 2016) with additive skip connections.

Detail branch. This branch has width 192, 12 Temporal NAF blocks, and four temporal residual fusion blocks with dilation rates $[1, 2, 1, 2]$. Its input comprises backbone features, $\mathbf{H}_x$, and seven groups of latent differences. For an input latent z, these are $z - \mathrm{Avg}_k(z)$ for $k \in \{3, 5, 9\}$, horizontal and vertical first differences, and temporal first and second differences. Spatial dimensions are padded to multiples of 16 and cropped back after reconstruction.

Fusion and quality heads. The features $\mathbf{F}_x$ are projected to 32 channels and concatenated with the input and the two reconstruction residuals. The fusion module has width 64 and four blocks. The quality head has width 48 and two blocks, receives detached features, and produces two channels. The dense prediction is

$$\begin{aligned} R &= \bar{z}_x + B + g_{\mathrm{det}} \odot D + \Delta_{\mathrm{fuse}}, \\ g_{\mathrm{det}} &= 2\,\mathrm{sigmoid}(g_{\mathrm{raw}}), \qquad \Delta_{\mathrm{fuse}} = 0.05\Delta_{\mathrm{raw}}. \end{aligned} \tag{A3}$$

Here, B and D are the backbone and detail residuals. The output scale of B is 0.107, and $D = 0.195(0.2 + 0.8\widetilde{\mathbf{H}}_x) \odot D_{\mathrm{raw}}$. With f indexing video frames, temporal smoothing uses

$$\widetilde{\mathbf{H}}_x[f] = 0.65\mathbf{H}_x[f] + 0.35\big(0.25\mathbf{H}_x[f-1] + 0.5\mathbf{H}_x[f] + 0.25\mathbf{H}_x[f+1]\big),$$

with replicated boundary frames. Equation (A3) omits the addition order used to preserve numerical rounding. For the two raw quality-head outputs $\mathbf{q}_{x,0}$ and $\mathbf{q}_{x,1}$, Wan uses $\mathbf{e}_x = 0.04\,\mathrm{softplus}(\mathbf{q}_{x,0})$ and LTX uses $\mathbf{e}_x = 0.10\,\mathrm{softplus}(\mathbf{q}_{x,0})$. Both use $\mathbf{u}_x = \mathrm{sigmoid}(\mathbf{q}_{x,1})$.

B.2 Reconstruction Risk and Candidate Set

Offline calibration. Following Sec. 3.4, we calibrate the learned quality estimates offline to obtain the reconstruction risk used by the router. Define $e_x(i) = [\mathcal{P}(\mathbf{e}_x)]_i$ and $u_x(i) = [\mathcal{P}(\mathbf{u}_x)]_i$. A monotone function κ maps $e_x(i)$ to local RGB error. For Wan, predicted errors from calibration trajectories are grouped into 20 equal-frequency bins. Sample-count-weighted isotonic regression fits the mean ground-truth error in each bin, with endpoints $(0, 0)$ and $(1, 1)$ and linear interpolation. These calibration prompts are disjoint from the final training and validation records in Sec. C.1. For LTX, predicted errors from validation-prompt trajectories are grouped into 24 equal-frequency bins; the monotone map uses each bin's median prediction and 80th-percentile ground-truth error, with endpoint clamping. Both maps are fixed before inference; routing evaluates κ on predicted errors without online RGB decoding.

Risk construction. At the entrance x, define

$$d_x(i) = \big[\mathcal{P}(\mathrm{mean}_C\,|\bar{z}_x - \bar{z}_{x-1}|)\big]_i, \qquad m_x(i) = \mathrm{clip}\left(\frac{d_x(i)}{\max(\mathrm{mean}_i\, d_x(i), 10^{-6})}, 0, 3\right).$$

The entrance corresponds to the tenth DiT evaluation. For the base reconstruction-error tolerance $\tau > 0$, the local tolerances are

$$\begin{aligned}\theta_x^{\mathrm{Wan}}(i) &= \frac{\tau(1-0.25b_x(i))(1+0.10(1-h_x(i)))}{1+0.20m_x(i)+0.15h_x(i)},\\ \theta_x^{\mathrm{LTX}}(i) &= \frac{\tau(1-0.25S_x(i))}{1+0.15m_x(i)+0.15h_x(i)}.\end{aligned} \tag{A4}$$

$$\begin{aligned}r_x(i) &= \frac{\kappa(e_x(i))+0.003m_x(i)}{\max(\theta_x(i),10^{-6})},\\ \rho_x(i) &= \max\left\{r_x(i), 0.75\max_{j\in\mathcal{N}_3(i)} r_x(j)\right\}.\end{aligned} \tag{A5}$$

Candidate set. Equation (A5) specifies the components of $\boldsymbol{\rho}_x$ in main-paper Eq. (5), where $\mathcal{N}_3(i)$ is a $3\times3\times3$ neighborhood. LTX uses $\Omega = \{i : \rho_x(i) > 1\}$. Wan additionally includes tokens on the 3×3 spatial morphological boundary of $\mathbf{b}_x$ whose risk exceeds 0.75. The set Ω is fixed after warmup, without a top-k or coverage cap.

B.3 Diffusion-State Alignment and Context

Dense context. At sampling step t, let σ_t denote the current noise level. The layer-wise K/V collections $\mathcal{B}_x$ and $\mathcal{B}_R$ come from the entrance DiT evaluation and an additional dense forward pass on R at the last nonzero noise level σ_R, respectively. The reference pass is omitted if Ω is empty. For each layer and conditioning branch, the alignment operator in main-paper Eq. (6) is

$$\begin{aligned}\widetilde{\mathcal{B}}_t &= \mathcal{A}(\mathcal{B}_R,\mathcal{B}_x;x,t) = \alpha_t\mathcal{B}_x + (1-\alpha_t)\mathcal{B}_R,\\ \alpha_t &= \mathrm{clip}\left(\frac{\sigma_t-\sigma_R}{\sigma_x-\sigma_R},0,1\right).\end{aligned} \tag{A6}$$

Let $\Delta\mathcal{B}_t$ store K/V residuals from earlier actual DiT observations, and let t^+ denote the state after processing the current observation. Unrefreshed locations use $\widetilde{\mathcal{B}}_t(i) + \Delta\mathcal{B}_t(i)$. Refreshed locations use the current K/V $\mathcal{B}_t^{\mathrm{DiT}}(i)$ and update

$$\Delta\mathcal{B}_{t^+}(i) = \mathcal{B}_t^{\mathrm{DiT}}(i) - \widetilde{\mathcal{B}}_t(i).$$

K/V residuals start at zero and remain unchanged when skipped. Queries, cross-attention, and MLPs are evaluated only for selected tokens. Self-attention reads the full spatiotemporal K/V grid, and RoPE retains the original grid coordinates.

B.4 Explicit DiT Computation and Observation Updates

Observation state. We use $\mathcal{H}_t = \{\mathcal{H}_t(i)\}_{i=1}^N$, where N is the number of DiT tokens. Let $b_t(i) < a_t(i) < t$ index the two most recent actual observations of token i. The state stores their clean predictions and acquisition indices, routing residual caches $\mathbf{d}_t^{(0)}(i)$ and $\mathbf{d}_t^{(1)}(i)$, stability evidence $E_t(i)$, innovation rate $\nu_t(i)$, and latest innovation magnitude $J_t(i)$. The solver separately maintains latent anchors.

Step-wise refresh. At the coded-segment entrance, the router Γ is configured with the auxiliary latent-risk map $\mathbf{u}_x$, the candidate set Ω, and the sampling schedule, which remain fixed throughout the segment. The tolerance τ governs the construction of $\boldsymbol{\rho}_x$ and Ω as described in Sec. B.2.

At each step t, define $\ell_t = \log(1-\sigma_t) - \log\sigma_t$ and $\Delta\ell_t(i) = \ell_t - \ell_{a_t(i)}$. The router $\Gamma(\mathbf{I}_x,\boldsymbol{\rho}_x,\mathcal{H}_t;\tau,\beta)$ in main-paper Eq. (7) combines importance, reconstruction risk, and observation history to determine the refresh demand. The auxiliary risk $u_x(i)$ contributes to the local scale $s_t(i)$, and the threshold $\beta > 0$ controls token refreshes. An entry $M_t(i) = 1$ requests explicit

Algorithm A1 Routing and actual-observation updates

Require: Fixed $\Omega, \mathbf{I}_x, \boldsymbol{\rho}_x, \mathbf{u}_x, R, \beta$; state $\mathcal{H}_t$ initialized as in Sec. B.4
function ROUTE($t, \mathcal{H}_t$)
  Compute $\Delta\ell_t$ from the latest actual observation indices.
  Compute D_t and $\mathbf{M}_t$ using Eq. (A7).
  if t is the last step **then**
    Set $M_t(i) \leftarrow \mathbf{1}[i \in \Omega]$.
  end if
  return $\mathbf{M}_t$
end function
function OBSERVE($t, \mathcal{H}_t, \mathbf{M}_t, \widehat{z}_t^{\mathrm{DiT}}$)
  for each token i with $M_t(i) = 1$ **do**
    Compute $\boldsymbol{\delta}_t(i) \leftarrow \widehat{z}_t^{\mathrm{DiT}}(i) - R(i)$.
    Extrapolate $\widehat{\boldsymbol{\delta}}_t(i)$ from the pre-update caches.
    Compute $J_{t^+}(i)$; update $\nu_{t^+}(i), E_{t^+}(i)$ using Eq. (A8).
    Roll the two clean observations, residual caches, and acquisition indices.
  end for
  Keep all other tokens' observations and derived caches unchanged.
  return the updated observation state
end function

computation of token i:

$$\begin{aligned} s_t(i) &= \mathrm{RMS}(\mathbf{d}_t^{(0)}(i)) + 0.25u_x(i) + 0.02, \\ n_t(i) &= \max_{j \in \mathcal{N}_3(i)} J_t(j) \exp(-\Delta\ell_t(j)), \\ D_t(i) &= I_x(i)[\Delta\ell_t(i)]^2 \left(\frac{\rho_x(i)}{1 + E_t(i)} + \frac{\nu_t(i) + n_t(i)}{s_t(i)} \right), \\ M_t(i) &= \mathbf{1}[i \in \Omega]\mathbf{1}[D_t(i) \geq \beta]. \end{aligned} \tag{A7}$$

At the final sampling step, the mask is set to $M_t(i) = \mathbf{1}[i \in \Omega]$ to refresh all candidate tokens regardless of their current demand. Only tokens with $M_t(i) = 1$ receive a fresh clean-state prediction $\widehat{z}_t^{\mathrm{DiT}}(i)$ and the observed correction $\boldsymbol{\delta}_t(i) = \widehat{z}_t^{\mathrm{DiT}}(i) - R(i)$ from main-paper Eq. (8). Before updating the caches, extrapolate

$$\widehat{\boldsymbol{\delta}}_t(i) = \mathbf{d}_t^{(0)}(i) + \frac{\Delta\ell_t(i)}{\ell_{a_t(i)} - \ell_{b_t(i)}} \big(\mathbf{d}_t^{(0)}(i) - \mathbf{d}_t^{(1)}(i) \big).$$

With $J_{t^+}(i) = \mathrm{RMS}(\boldsymbol{\delta}_t(i) - \widehat{\boldsymbol{\delta}}_t(i))$, the derived state is updated as

$$\begin{aligned} \nu_{t^+}(i) &= e^{-\Delta\ell_t(i)} \nu_t(i) + (1 - e^{-\Delta\ell_t(i)}) \frac{J_{t^+}(i)}{[\Delta\ell_t(i)]^2}, \\ E_{t^+}(i) &= E_t(i) + \Delta\ell_t(i) \exp\left(-\frac{\mathrm{RMS}(\boldsymbol{\delta}_t(i) - \mathbf{d}_t^{(0)}(i))}{0.25u_x(i) + 0.02} \right). \end{aligned} \tag{A8}$$

We then set $\mathbf{d}_{t^+}^{(1)}(i) = \mathbf{d}_t^{(0)}(i)$, $\mathbf{d}_{t^+}^{(0)}(i) = \boldsymbol{\delta}_t(i)$, and $(a_{t^+}(i), b_{t^+}(i)) = (t, a_t(i))$. When a token is skipped, its observations and derived caches remain unchanged; only its current refresh demand is recomputed.

Initially, $E_t(i) = \nu_t(i) = J_t(i) = 0$. Wan initializes its routing residual caches to zero, whereas LTX initializes them from the last two warmup observations minus $R(i)$. Both solvers retain the two actual warmup observations. Zero-initialized caches and reconstructed values are not inserted as new observations.

Inference settings. Both backbones use 50 sampling steps, 10 warmup steps, and $\tau = 0.010$. The Low, Mid, and High settings use $\beta = 0.1$, 1.0, and 2.0, respectively. All candidate tokens are forcibly refreshed at the last step.

Algorithm A2 Coded-computation inference

Require: Frozen DiT F_θ, text condition c, initial noise, a 50-step schedule, and τ, β
Ensure: Exit latent z_y and compute code $\mathcal{C}_{x\to y}$

```
Run 10 dense warmup steps; retain two clean predictions, z_{x+}, solver history, and B_x.
Compute V_x, S_x, I_x, R, ρ_x; construct Ω as in Sec. B.2.
Initialize H_t, the empty sequence C_{x→y}, and zero K/V residuals.
if Ω ≠ ∅ then
    Extract B_R by a reference forward pass on R.
end if
for each remaining sampling step t do
    M_t ← ROUTE(t, H_t)                                   ▷ Algorithm A1
    if Σ_i M_t(i) > 0 then
        Align references using Eq. (A6) and add K/V residuals.
        Evaluate F_θ only at selected tokens.
        H_{t+} ← OBSERVE(t, H_t, M_t, ẑ_t^DiT)
        Update the K/V residuals at the evaluated locations.
    end if
    Compose the next dense state using the backbone solver and Eq. (A9).
    Append M_t to C_{x→y}.
end for
return z_y, C_{x→y}; decode z_y with the VAE decoder D_VAE.
```

B.5 State Composition and Complete Inference

Latent states. Let z_t be the complete latent at the current step, and z_t^{hist} the candidate-region state advanced from actual observation history. The noisy state z_{x^+} and noise level σ_{x^+} are recorded after the entrance solver update; they are distinct from the entrance clean prediction $\bar{z}_x$. Directly reconstructed and candidate regions are composed as

$$\begin{aligned} z_t^{\text{rec}} &= R + \frac{\sigma_t}{\sigma_{x^+}}(z_{x^+} - R), \\ z_t &= \mathbf{1}_\Omega \odot z_t^{\text{hist}} + (1 - \mathbf{1}_\Omega) \odot z_t^{\text{rec}}. \end{aligned} \tag{A9}$$

Masks are broadcast to the corresponding latent patches. Wan advances z_t^{hist} with token-wise second-order UniPC (`bh2`, `predict-x0`), retaining each token's own observation times and two actual clean predictions. LTX uses native RF Euler updates at refreshed tokens. At skipped tokens, it integrates the linear trend of the two actual clean predictions in the ℓ domain over the native Euler time grid. The composed state at the coded-segment exit y is z_y. When $\sigma_y = 0$, the directly reconstructed region equals R.

For LTX token i, write $a = a_t(i)$ and $b = b_t(i)$, and anchor the state after the native Euler update at step a. For skipped steps $k \geq a + 1$, the equivalent recurrence is

$$\begin{aligned} \tilde{z}_k^{\text{clean}}(i) &= \hat{z}_a^{\text{DiT}}(i) + \frac{\ell_k - \ell_a}{\max(\ell_a - \ell_b, 10^{-6})}\big(\hat{z}_a^{\text{DiT}}(i) - \hat{z}_b^{\text{DiT}}(i)\big), \\ z_{k+1}^{\text{hist}}(i) &= \gamma_k z_k^{\text{hist}}(i) + (1 - \gamma_k)\tilde{z}_k^{\text{clean}}(i), \qquad \gamma_k = \sigma_{k+1}/\sigma_k. \end{aligned} \tag{A10}$$

The implementation precomputes the recurrence coefficients to obtain a target-step state directly from its anchor. LTX clamps both σ and $1-\sigma$ from below by 10^{-6} when computing ℓ. Reconstructed values never enter the actual-observation history.

Collecting $\mathbf{M}_t$ in sampling order yields the compute code $\mathcal{C}_{x\to y}$ in main-paper Eq. (2). Algorithm A2 gives the complete procedure.

Implementation settings. K/V caches use BF16 and are scoped to the current request. Fused operators perform K/V interpolation, residual addition, and updates at the current locations. Wan batches CFG branches and caches text K/V. LTX shares reusable CFG/STG prefixes and retains full-grid outputs for guidance rescaling. After warmup, LTX enables fused RMS normalization, whose floating-point reduction order differs from the native implementation. If only one token is selected, one neighboring support token is added.

C Training Settings and Objectives

C.1 Training Settings

Wan and LTX follow the same training procedure in Sec. 3.4, with frozen auxiliary predictors and a four-group objective for jointly training the reconstruction module and quality heads. Clip lengths here refer to teacher training data; evaluation lengths follow Sec. 4.1.

Teacher trajectories. We collect teacher trajectories by randomly sampling prompts. The frozen native model performs 50 dense sampling steps for each prompt and random seed. Inputs are clean-state predictions along the trajectory, and the target $R^\star$ is its terminal latent. Warmup clean predictions and noisy states after solver updates are stored separately.

Wan settings. Three seeds per randomly sampled prompt generate 41-frame teacher videos at 480×832, with the step-10 prediction used as input. The visual and geometry branches are frozen, while the RGB-error head is trained with the reconstruction module. AdamW (Loshchilov & Hutter, 2019) uses fixed learning rates of 3×10^{-5} for the backbone and 10^{-4} for the remaining trainable parameters.

LTX settings. Teacher videos generated from randomly sampled prompts contain 17 frames at 704×1216, yielding a $3 \times 22 \times 38$ latent grid. Inputs use the step-10 prediction with probability 0.8 and steps 11, 15, 20, 25, and 30 with probability 0.04 each. Reconstruction training uses AdamW with cosine decay from 10^{-4} to 10^{-5}. The visual branch and reconstruction module are then frozen, and the RGB-error head is trained separately for 3,000 updates with batch size 8 and an AdamW learning rate of 10^{-4}.

Shared settings. Final reconstruction training uses BF16, gradient norm clipping at 1, and weight decay 10^{-4} for both backbones. The separate LTX error-head stage uses the same settings.

Auxiliary-branch training. The visual branch is first distilled from decoded step-10 images, using 40 AlexNet (Krizhevsky et al., 2012) feature channels, three RGB channels, three adjacent-frame difference channels, and a detail map. The optimizer is AdamW with learning rate 4×10^{-4} and weight decay 0.01. The geometry branch is finally fine-tuned on YOLOE (Wang et al., 2025) masks of step-10 images. Its objective is foreground/background-balanced BCE plus Dice loss.

Initialization and checkpoint selection. Wan initializes the final reconstruction stage from a pre-trained reconstruction checkpoint. LTX first transfers shape-compatible Wan hidden layers and then adapts the 128-channel interfaces. Wan selects checkpoints by validation RGB L1 + 2 MSE + 0.1 LPIPS + 0.5 temporal error. LTX selects its reconstruction and quality-head checkpoints by latent MSE and RGB-error MAE, respectively, yielding checkpoints at updates 4,750 and 2,800.

C.2 Training Objectives

Wan loss definitions. The following terms instantiate the four loss groups in main-paper Eq. (9). Let $E_z = R - R^\star$ and $E_{\mathrm{rgb}} = \mathcal{D}_{\mathrm{VAE}}(R) - \mathcal{D}_{\mathrm{VAE}}(R^\star)$, with RGB values in $[0, 1]$. Wan uses $w = 1 + 0.5\mathbf{G}_x^{\mathrm{lat}}$ and $\langle a \rangle_w = \sum wa/(C \sum w)$, where C is the channel count of the error tensor. The operators $\nabla_f, \nabla_h, \nabla_w$ are first differences along frames, height, and width. The following are the already weighted terms $\lambda_k \mathcal{L}_k$, whose sum gives $\mathcal{L}$ without additional weighting. The indicator $\chi_{\mathrm{rgb}} \in \{0, 1\}$ enables RGB supervision every other minibatch, and sg denotes stop-gradient.

Reconstruction term.

$$\begin{aligned}\lambda_{\mathrm{rec}}\mathcal{L}_{\mathrm{rec}} = {} & 0.7\langle |E_z| \rangle_w + 0.15\langle E_z^2 \rangle_w + 0.05\,\mathrm{mean}\,|\nabla_w E_z| \\ & + 0.002\,\mathrm{mean}\,\Delta_{\mathrm{fuse}}^2 \\ & + \chi_{\mathrm{rgb}}\big(3\langle |E_{\mathrm{rgb}}| \rangle_w + 10\langle E_{\mathrm{rgb}}^2 \rangle_w + 2\mathcal{L}_{\mathrm{grad}}\big).\end{aligned}$$

Temporal term.

$$\lambda_{\mathrm{tem}}\mathcal{L}_{\mathrm{tem}} = 0.2\langle |\nabla_f E_z| \rangle_w + 2\chi_{\mathrm{rgb}}\langle |\nabla_f E_{\mathrm{rgb}}| \rangle_w.$$

Perceptual term.

$$\lambda_{\mathrm{per}}\mathcal{L}_{\mathrm{per}} = \chi_{\mathrm{rgb}}\{0.25\mathcal{L}_{\mathrm{LPIPS}} + 0.15(1 - \mathrm{SSIM})\}.$$

Quality-estimation term.

$$\lambda_{\mathrm{qua}}\mathcal{L}_{\mathrm{qua}} = 0.03\,\mathrm{BCE}\left(\mathbf{u}_x, \mathrm{clip}\left(\frac{\mathrm{mean}_C\,|\,\mathrm{sg}(E_z)|}{0.25}, 0, 1\right)\right) + 0.1\chi_{\mathrm{rgb}}\big\{\mathrm{SmoothL1}(\mathbf{e}_x/0.04, \mathbf{q}/0.04) + 0.15\mathcal{L}_{\mathrm{cls}} + 0.10\mathcal{L}_{\mathrm{rank}}\big\}.$$

RGB and quality supervision. RGB supervision uses 24×32 latent spatial crops with all temporal frames, alternating between subject-centered and random locations. The loss $\mathcal{L}_{\mathrm{grad}}$ averages horizontal and vertical RGB gradient errors; SSIM (Wang et al., 2004) uses a 5×5 window, and LPIPS (Zhang et al., 2018) uses AlexNet. The quality label $\mathbf{q}$ is detached local RGB MAE, aggregated according to the VAE layout: the first frame separately, then groups of four frames and 8×8 pixels. It is distinct from the raw head outputs $\mathbf{q}_{x,0}, \mathbf{q}_{x,1}$.

The classification term $\mathcal{L}_{\mathrm{cls}}$ uses thresholds $\eta \in \{0.005, 0.010, 0.020, 0.040\}$. It computes BCE between logits $(\mathbf{e}_x - \eta)/0.005$ and labels $\mathbf{1}[\mathbf{q} > \eta]$, averaged over positions and thresholds. For $\mathcal{L}_{\mathrm{rank}}$, flattened labels are paired by a circular shift of 137 positions, retaining only pairs whose label difference exceeds 0.005. The loss averages

$$\mathrm{softplus}\left(-\frac{(\mathbf{e}_x[p] - \mathbf{e}_x[\pi(p)])\,\mathrm{sign}(\mathbf{q}[p] - \mathbf{q}[\pi(p)])}{0.01}\right),$$

where $\pi(p)$ is the shifted partner of position p. The loss is zero if no valid pair remains.

LTX adaptation. LTX uses the latent objective below during reconstruction training. Let $\mathrm{SL1}_b$ denote SmoothL1 with transition parameter b, let $\mathbf{a}_z = \mathrm{mean}_C\,|\,\mathrm{sg}(E_z)|$, and let $\mathbf{d}_z$ be the channel RMS of the difference between the target latent and its 3×3 spatial average:

$$\begin{aligned}\mathcal{L}_{\mathrm{LTX}} = {} & \mathrm{mean}\,E_z^2 + \frac{0.2}{3}\sum_{j\in\{f,h,w\}} \mathrm{mean}(\nabla_j E_z)^2 + 0.05\,\mathrm{mean}\,|E_z| \\ & + 0.3\,\mathrm{SL1}_{0.1}(\mathbf{e}_x, \mathbf{a}_z) + 0.01\,\mathrm{BCE}(\mathbf{u}_x, \mathbf{1}[\mathbf{a}_z > 0.12]) \\ & + 0.02\,\mathrm{MSE}(\mathbf{H}_x, \mathbf{d}_z/(\mathbf{d}_z + 0.15)).\end{aligned} \tag{A11}$$

This latent-space objective specifies the LTX training adaptation of the main-paper framework. It replaces Wan's RGB/LPIPS/SSIM loss combination during reconstruction training, and its final term adapts the visual branch. At this stage, $\mathbf{e}_x$ predicts latent error. The subsequent quality-head-only stage uses

$$\mathrm{SL1}_{0.01}(\mathbf{e}_x, \mathbf{q}) + 0.01\,\mathrm{BCE}(\mathbf{u}_x, \mathbf{1}[\mathbf{a}_z > 0.12]).$$

RGB labels aggregate the first frame separately, then groups of eight frames and 32×32 pixels. Prediction and target share the decoding noise: the decoder input is $0.975z + 0.025\varepsilon$ at decoding time 0.05, where ε is Gaussian noise.

D Background Completion and Replacement

These settings specify the optional editing examples in Sec. 4.4; the acceleration comparisons follow Sec. 4.1.

Completion input and masks. We decode Wan's step-10 reconstructed prediction as $Y_R = \mathcal{D}_{\mathrm{VAE}}(R)$, using 41 frames at 480×832. YOLOE (Wang et al., 2025) initializes subject masks and SAM2.1 (Ravi et al., 2025) propagates them bidirectionally on both Y_R and the original accelerated output. The union of the two mask sequences is processed with a three-frame temporal maximum filter and 31×31 spatial dilation. The removal region also covers subject-associated effects such as shadows and may be rectangularized to reduce contour remnants.

External completion. LaMa (Suvorov et al., 2022) or Wan2.1-VACE-1.3B (Jiang et al., 2025) supplies completion through a common video-and-hole-mask interface. Let Q be the binary hole mask, $\widetilde{Q}$ its Gaussian-feathered version, and $\widehat{Y}_{\mathrm{bg}}$ the completion prediction. The basic blend with the known background is

$$Y_{\mathrm{fill}} = (1 - \widetilde{Q}) \odot Y_R + \widetilde{Q} \odot \widehat{Y}_{\mathrm{bg}}. \tag{A12}$$

LaMa's frame-wise outputs are temporally smoothed with a standard deviation of 1.2 frames. VACE completion uses 50 steps, shift 5, CFG 7, and conditioning strength 1. Boundary processing includes mask feathering and Poisson gradient-domain blending.

Background generation and insertion. For replacement, Depth Anything V2 Small extracts frame-wise relative depth from the completed video. Depth is temporally smoothed with a standard deviation of 0.8 frames and normalized using the 0.5th and 99.5th percentiles over the entire video. The depth video and scene text condition are provided to VACE, using 50 steps, shift 5, CFG 5, and conditioning strength 1. The resulting background video is encoded by the original Wan VAE as R_{bg}. The source foreground mask is trilinearly resized to the $11 \times 60 \times 104$ latent grid, followed by a $3 \times 3 \times 3$ maximum filter and spatial mean feathering to obtain A_{fg}. The insertion interface is

$$R_{\mathrm{edit}} = A_{\mathrm{fg}} \odot R + (1 - A_{\mathrm{fg}}) \odot R_{\mathrm{bg}}. \tag{A13}$$

State handoff and foreground compositing. The background reference is forward-noised as $z_{\mathrm{bg}}(\sigma) = (1 - \sigma)R_{\mathrm{bg}} + \sigma\varepsilon$. Compatible background K/V references are extracted at the noise level of the tenth DiT call. A soft mask on the token grid blends the entrance K/V references; the scene text condition is switched, and background states at the corresponding noise levels are injected during coded reconstruction over steps 11–50. Final RGB compositing retains the foreground and coordinates of the source accelerated video. Only temporally smoothed, low-frequency color gains from the continued generation are applied, bounded by $\exp(\pm 0.13)$. External segmentation, completion, background generation, reference-state preparation, and compositing are timed separately.

E Additional Experimental Details and VBench Results

We follow the evaluation protocol in Section 4.1. This section specifies the baseline operating points and LTX adaptations, and reports the VBench dimensions omitted from Table 1.

E.1 Baseline Configurations and LTX Adaptation

Alignment with official implementations. We follow the released baseline implementations and recommended configurations as closely as permitted by the evaluated backbone, retaining their method-specific cache criteria, thresholds, warm-up and refresh schedules, and sensitivity data. For LTX, the official caching logic is integrated with the shared 0.9.6 backbone and its native conditioning branches and output path. The settings and model-specific adaptations below define the configurations used for the reported comparisons.

Operating points. All methods retain the backbone's native weights and generation conditions. In addition to the settings in Section 4.1, Wan uses a timestep shift of 2.0 and 16 fps. LTX uses the 0.9.6 checkpoint, 30 fps, spatiotemporal skip guidance (STG) (Hyung et al., 2025) scale 1.0, guidance rescaling 0.7, and STG block 19 (zero-indexed). Dynamic flow scaling is enabled with flow scale 0.1; the VAE decoding timestep and noise scale are 0.05 and 0.025.

TeaCache (Liu et al., 2025a). The thresholds are 0.08 for Wan and 0.05 for LTX. Wan disables the additional retention-step option (`use_ret_steps=False`). The LTX adapter retains the TeaCache4LTX change signal from the modulated first-block input, polynomial error estimate, and accumulated-error gate. Following its official schedule, LTX performs full computation at the first and last denoising steps; intermediate steps follow the accumulated-error gate with threshold 0.05. Residuals are cached after output normalization and modulation, before the output projection. A reused residual is added to the current projected input, followed by the native output projection.

NaviCache (Lv et al., 2026). Both backbones use threshold 0.04, with process and measurement noise set to 0.05; Wan uses `align_steps=10`. The LTX adapter preserves the official Wan implementation's input/output change ratio, state estimation, and accumulated-error criterion. Sequential CFG calls are mapped to LTX's batched negative, positive, and STG branches. The positive branch determines whether to refresh, while each branch maintains its own raw Transformer input-to-output residual.

SenCache (Haghighi & Alahi, 2026). Wan uses start/main thresholds $0.045/2.0$, at most three consecutive skips, and the official `sensitivity_wan21.npz`. LTX follows the official two-stage threshold schedule: the unscaled threshold is 0.01 for the first 10 denoising steps and 0.7

for the remaining 40 steps. The first and last steps are fully computed, and $K = 4$ bounds consecutive reuse. Its adapter retains the official sensitivity criterion and `sensitivity_ltx.npz`, measured for LTX 0.9.1 and reused with the 0.9.6 backbone. Native noise levels are multiplied by 1000 only for sensitivity lookup and the corresponding time differences; the official threshold scale is 1015.9685. Latent-difference norms are accumulated in FP32. The residual boundary is after normalization/modulation and before the output projection.

ProfilingDiT (Ma et al., 2025b). We retain the upstream block and consecutive-block-group residual caching logic with `step_start=6` and `step_interval=6`. For LTX, the upstream $28/40$ cached-layer fraction is mapped to $20/28$ layers, selected by the existing foreground-attention ranking obtained with RGB foreground segmentation. The zero-indexed cached layers are $\{4, 7\} \cup \{10, \ldots, 27\}$. The last step is not additionally forced to be dense. This is an adaptation of the upstream caching logic; the upstream implementation does not provide a native LTX version.

LTX execution. The adapters retain the native scheduler, conditioning branches, and output projection, and reset caches for each video. Full evaluations use the frozen native forward pass. Our reconstruction network and compute-code router are used only by our method. The reported LTX TeaCache and SenCache results use the official cache schedules specified above, with the same backbone and generation conditions as the other methods.

E.2 VBench Dimension Scores

Table 4 reports both backbones and complements the aggregate scores in Table 1. Individual dimensions are the evaluator's raw scores multiplied by 100. Quality and semantic aggregates use the official VBench normalization and weighting (Huang et al., 2024) rather than an arithmetic mean of the displayed dimensions.

Wan. Across Low, Mid, and High, quality aggregates remain within 0.04 points of the native score. At High, subject consistency, background consistency, and motion smoothness are 95.72, 97.28, and 98.61, compared with 95.68, 97.27, and 98.59 for native inference. Dynamic degree is 55.09 versus 54.17, indicating that acceleration does not coincide with a lower dynamic-degree score. At Low, scene and temporal-style scores match native inference, and overall consistency differs by only 0.01 points.

LTX. In the current results, Low obtains quality/semantic aggregates of $83.10/57.21$, compared with native scores of $83.09/57.07$. High retains temporal-flickering and motion-smoothness scores of 99.81 and 99.33, identical to native inference at the reported precision, while its dynamic-degree score is 42.59 versus 42.13. These dimensions supplement the paired fidelity metrics by showing which generation properties are retained under selective execution.

F Additional Qualitative Comparisons

Figures 8 and 9 extend Fig. 4 with eight examples per backbone. On Wan, jellyfish contours, architectural columns, animal outlines, water droplets, and fur illustrate the preservation of both structure and local appearance. On LTX, leaf edges, guitar strings, hair, window grids, and patterned surfaces provide complementary detail comparisons. Low more closely matches the local native appearance, while High retains the main structures with fewer explicit evaluations.

G Perceptual-Semantic Encoder Ablation Details

Shared ablation protocol. Tables 2 and 3 evaluate 944 prompts with seeds 0, 42, and 114514, matching the Wan main-comparison prompts and seeds in Sec. 4.1. They examine encoder-guided computation allocation and the three decoder operations, respectively. We generate 81 frames at 832×480 and 16 fps, with 50 denoising steps, CFG 5.0, shift 2.0, ten dense warm-up steps, $\tau = 0.01$, and $\beta = 2.0$. Both tables use the same full-method outputs and native Wan references, and report mean LPIPS, SSIM, and PSNR over the evaluated prompt–seed pairs.

Ablation target. The encoder in Section 3.2 provides visual and semantic cues for constructing the compute code. These ablations intervene in how its outputs guide execution selection: Random and History coverage provide content-neutral controls, ROI random provides a fixed spatial prior,

Table 4: **VBench dimension scores on Wan2.1-1.3B and LTX-Video-2B (%).** Operating points match Table 1; the overall score is reported there. Individual dimensions and normalized aggregates are shown separately.

Dimension	Native	TeaCache	NaviCache	SenCache	ProfilingDiT	Ours		
						Low	Mid	High
Wan2.1-1.3B								
Quality aggregate	83.17	83.46	83.10	83.40	82.95	83.21	83.19	83.17
Semantic aggregate	65.71	65.37	65.88	64.74	64.93	65.39	64.90	64.66
Subject consistency	95.68	95.69	95.53	95.77	95.62	95.68	95.73	95.72
Background consistency	97.27	98.02	97.68	97.96	97.78	97.27	97.33	97.28
Temporal flickering	99.52	99.39	99.34	99.35	99.26	99.51	99.47	99.44
Motion smoothness	98.59	98.59	98.59	98.61	98.58	98.59	98.61	98.61
Dynamic degree	54.17	54.63	52.78	54.17	52.78	54.63	54.63	55.09
Aesthetic quality	60.81	62.03	61.07	61.97	61.17	60.84	60.88	60.84
Imaging quality	66.74	66.45	66.82	66.44	65.77	66.75	66.50	66.36
Object class	79.40	79.40	78.48	77.90	76.93	78.88	79.11	78.93
Multiple objects	55.36	55.18	54.78	54.78	53.76	54.88	54.37	54.32
Human action	72.67	71.33	73.33	73.00	71.67	72.00	72.00	73.00
Color	91.73	91.39	91.18	90.01	90.95	90.51	89.36	88.30
Spatial relationship	66.69	67.88	71.10	63.79	67.16	66.76	64.99	63.28
Scene	20.86	19.60	20.49	20.16	21.15	20.86	20.59	20.59
Appearance style	20.71	20.52	20.56	20.47	20.47	20.68	20.63	20.67
Temporal style	22.84	22.74	22.67	22.65	22.49	22.84	22.67	22.60
Overall consistency	23.66	23.68	23.64	23.61	23.58	23.65	23.56	23.55
LTX-Video-2B								
Quality aggregate	83.09	82.87	83.08	82.60	82.32	83.10	83.06	83.03
Semantic aggregate	57.07	56.42	57.36	57.17	56.11	57.21	56.90	56.85
Subject consistency	96.62	96.53	96.61	96.54	96.55	96.63	96.63	96.63
Background consistency	98.48	98.48	98.50	98.46	98.51	98.51	98.52	98.53
Temporal flickering	99.81	99.81	99.81	99.81	99.79	99.81	99.81	99.81
Motion smoothness	99.33	99.35	99.33	99.40	99.37	99.33	99.33	99.33
Dynamic degree	42.13	41.20	42.59	37.50	40.28	42.13	42.59	42.59
Aesthetic quality	56.14	55.91	56.06	55.83	55.53	56.18	55.91	55.84
Imaging quality	70.85	70.17	70.59	70.13	67.39	70.83	70.56	70.49
Object class	51.61	48.92	51.42	51.16	49.31	50.79	50.40	49.76
Multiple objects	36.31	33.41	36.69	36.71	33.38	36.46	36.08	35.72
Human action	63.67	62.67	63.33	63.33	59.33	63.67	63.00	62.33
Color	90.94	91.84	93.05	91.88	92.44	92.38	92.35	92.42
Spatial relationship	64.69	63.91	65.16	66.71	65.08	65.16	64.53	64.77
Scene	6.73	7.17	6.88	5.64	6.95	6.90	6.40	7.27
Appearance style	22.30	22.33	22.28	22.13	22.18	22.29	22.25	22.24
Temporal style	21.22	21.18	21.24	21.28	21.16	21.19	21.24	21.22
Overall consistency	22.52	22.55	22.53	22.55	22.29	22.50	22.48	22.47

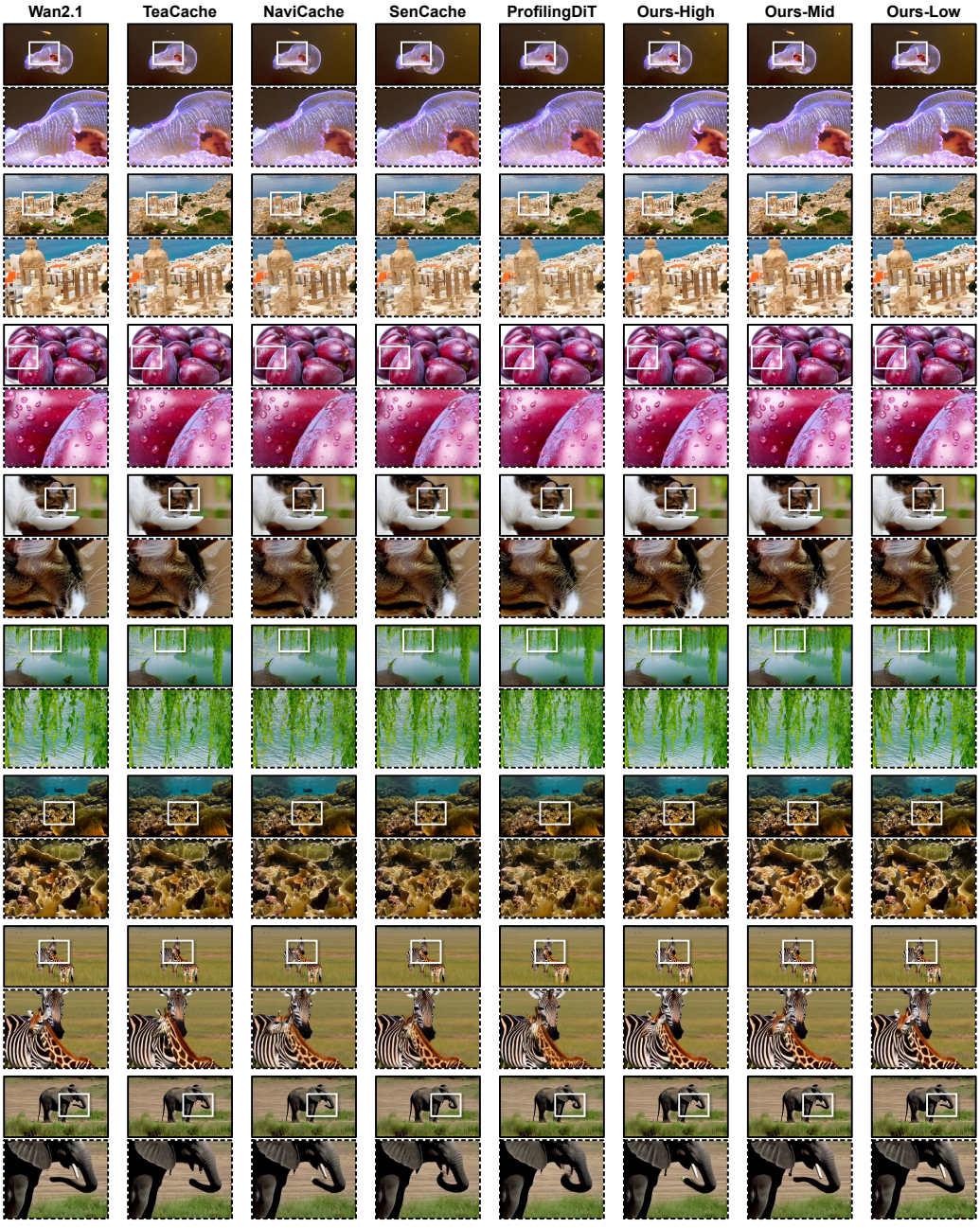

Figure 8: **Additional qualitative comparisons on Wan.** From left to right: Native, TeaCache, NaviCache, SenCache, ProfilingDiT, Ours–High, and Ours–Low. Each row shows a generated frame and enlarged local details under matched prompts and random seeds.

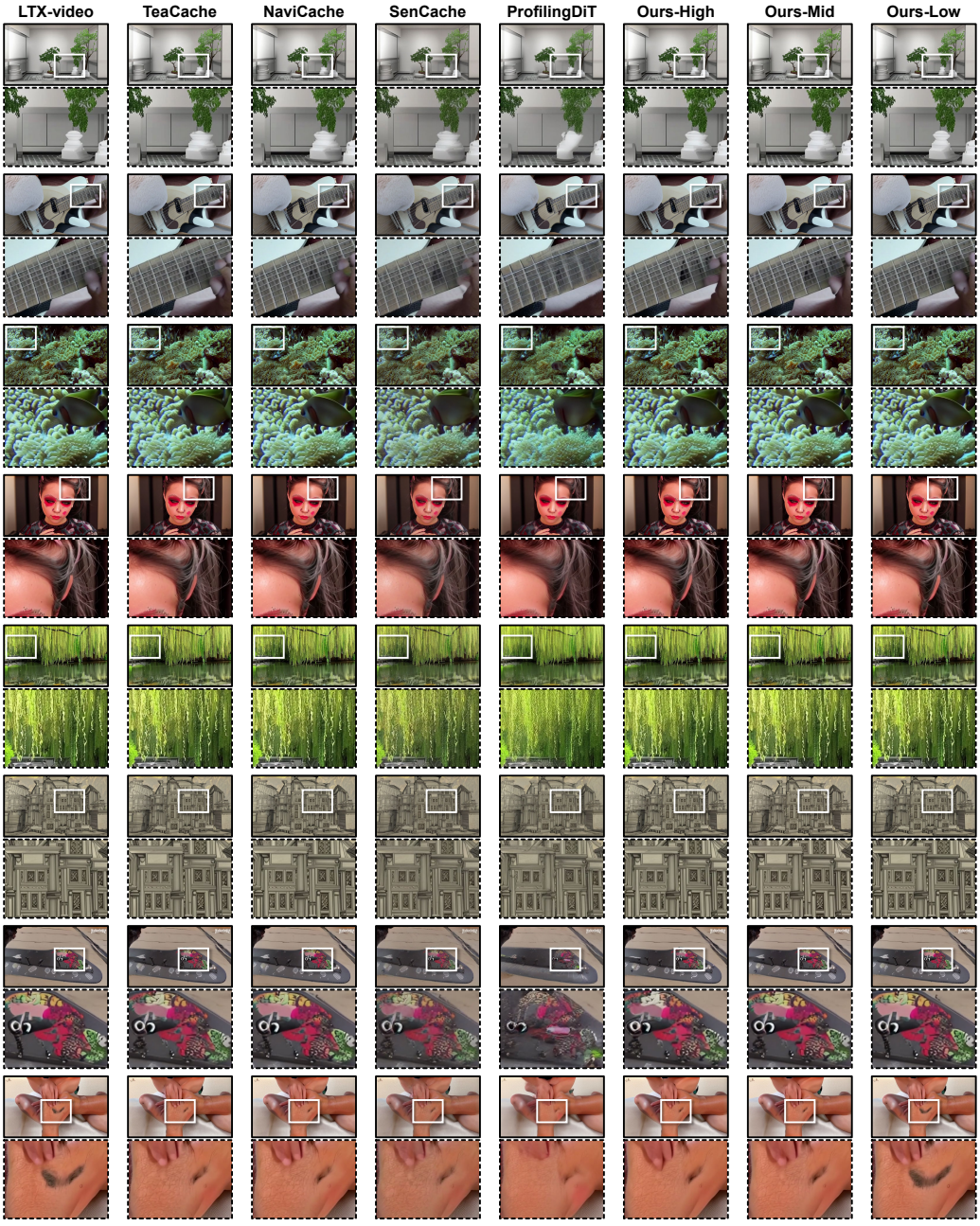

Figure 9: **Additional qualitative comparisons on LTX.** From left to right: Native, TeaCache, NaviCache, SenCache, ProfilingDiT, Ours–High, and Ours–Low. Each row shows a generated frame and enlarged local details under matched prompts and random seeds.

and Visual/Semantic variants use the corresponding encoder cue. Ours retains the complete adaptive allocation rule, which integrates both continuous cues with reconstruction risk and observation feedback. The cue extractors and generative reconstruction module are shared across variants.

Matched execution budgets. Encoder variants share the candidate set Ω, reconstruction network, and alignment. At every step, each selector chooses exactly $n_t = \sum_i M_t(i)$ candidates, with n_t taken from the full method. Only the selected locations differ. All new observations come from actual evaluations of the same frozen DiT.

Random and region-prioritized selection. Uniform random samples within Ω. ROI random prioritizes the central half of both spatial axes, shared across video time. Semantic random prioritizes the binary semantic region intersected with Ω. Visual random ranks candidates by detail response and takes a region with the same number of tokens as the semantic region. Region-prioritized variants shuffle tokens within and outside the preferred region, filling any remaining quota from outside it.

History coverage. Coverage ranks candidates by the interval since their last actual DiT observation, measured in the sampler's noise-time coordinate. The neutral variant breaks ties randomly. Visual and Semantic coverage use their size-matched binary regions only to break history ties, followed by random tie-breaking. Thus, content preference never displaces a token with an older observation. The final step refreshes the shared candidate set for every selector.

Contribution of the encoder cues. History coverage establishes a content-neutral scheduling baseline: it reduces LPIPS from 0.0810 to 0.0653 relative to uniform random sampling and raises PSNR from 26.68 to 28.09 dB. Relative to this baseline, using the encoder's visual cue lowers LPIPS to 0.0521, raises SSIM from 0.8822 to 0.9011, and improves PSNR by 1.14 dB. Using its semantic cue gives 0.0521 LPIPS, 0.9007 SSIM, and a 1.07 dB PSNR gain. Both guided-coverage variants reduce LPIPS by 20.2%. Because these comparisons retain the same observation-age priority and update budget, they attribute the additional selection benefit to information supplied by each encoder cue.

Role in the complete allocation rule. Region-prioritized random variants underperform uniform random sampling, indicating that spatial preference alone does not reliably identify which tokens currently need fresh computation. The complete rule combines continuous perceptual-semantic guidance with reconstruction risk and observation feedback. It obtains the best values of all three fidelity metrics: 0.0330 LPIPS, 0.9279 SSIM, and 31.27 dB PSNR. This reduces LPIPS by 36.7% relative to either guided-coverage variant and improves PSNR by 2.04 dB over Visual coverage and 2.11 dB over Semantic coverage. The cue-controlled comparisons support the encoder's role in guiding computation locations at a fixed update budget. The comparison with Ours evaluates their integration with reconstruction risk and observation feedback in the full allocation rule, matching the analysis in Sec. 4.3.

H Code-Aware Generative Decoder Ablation Details

The decoder variants use the protocol in Section G and replay the same complete execution-mask sequence. Consequently, each step acquires fresh observations at identical token locations. The three switches in Table 3 intervene directly in the original decoder's state reconstruction, contextual reference construction, and reference alignment. Their eight combinations include the complete decoder and allow each operation to be examined with the other two retained.

State reconstruction. This switch controls the clean-state endpoint used for the directly reconstructed region in Section B.5. When enabled, that endpoint is the learned prediction R; when disabled, it is a fixed entrance-history extrapolation P. The solver and noisy-state composition remain unchanged.

Contextual references. This switch controls the content used to construct the dense reference K/V in Section B.3. It uses R when enabled and the same P when disabled. The full spatiotemporal context is retained in both cases, isolating the value of learned reference content.

Diffusion-state alignment. This switch controls the noise-dependent interpolation between entrance and predicted-reference K/V in Section B.3. Disabling it sets the entrance interpolation coefficient to zero, leaving the predicted-endpoint reference without noise-dependent interpolation. This intervention concerns contextual K/V only; the noisy latent-state composition in Section B.5 remains enabled throughout.

Fixed extrapolation. Let x^- and x denote the last two actual DiT observations before coded execution, with clean predictions $\bar{z}_{x^-}, \bar{z}_x$ and noise levels σ_{x^-}, σ_x. The replacement endpoint is

$$P = \bar{z}_x + \frac{-\sigma_x}{\sigma_x - \sigma_{x^-}} \big(\bar{z}_x - \bar{z}_{x^-}\big).$$

It is fixed once at the entrance and requires no additional DiT call. Using the same substitute for both switches makes their meanings consistent across the factorial ablation.

Contribution of the decoder operations. A6 versus Ours isolates the learned state-reconstruction path: using R instead of the fixed endpoint reduces LPIPS from 0.1456 to 0.0330 and improves PSNR by 4.75 dB. This measures the value of learned content for directly reconstructing uncomputed regions. A5 versus Ours isolates learned contextual content: using R to construct reference content reduces LPIPS from 0.1046 to 0.0330 and improves PSNR by 6.59 dB, demonstrating its contribution to the explicitly evaluated tokens through their dense context. A3 versus Ours isolates diffusion-state alignment: restoring noise-dependent reference alignment reduces LPIPS from 0.1094 to 0.0330 and improves PSNR by 6.16 dB, showing the benefit of matching reference content to the current diffusion state. Each comparison retains the other two operations and identical execution masks. The observed differences therefore identify the individual roles of the original decoder's three paths under the same explicit-computation schedule.

I Further Analysis of Routing Sensitivity

Sweep protocol. Figure 5 uses the prompt "A small blue bird turns its head on a branch among dense green leaves, soft daylight, a fixed camera." The random seed is 2026091401. Videos contain 41 frames at 832×480, with 50 denoising steps, CFG 5.0, shift 2.0, and ten dense warm-up steps. The 21 configurations combine $\tau \in \{0.005, 0.0075, 0.01, 0.0125, 0.015, 0.0175, 0.02\}$ with $\beta \in \{0.1, 1.0, 2.0\}$. Latency is the median of three measurements after one warm-up run.

Candidate support versus actual execution. At $\tau = 0.01$, all three β values share a candidate set covering 99.02% of tokens. Nevertheless, their mean execution ratios over the forty coded steps are 79.84%, 28.13%, and 21.03% for Low, Mid, and High. A large candidate set therefore does not imply dense execution: β determines when a candidate needs a new observation. The corresponding latency reductions are 37.8% from Low to Mid and a further 9.1% from Mid to High.

Effect of the reconstruction tolerance. Increasing τ from 0.005 to 0.02 reduces candidate coverage from 100% to 70.28% in this example. Thus, τ changes the support requiring explicit tracking, whereas β controls refresh demand along the trajectory. Both parameters operate with the same trained network. For clarity, coded-step execution is normalized by $40N$, where N is the full token count; including ten dense steps gives an overall ratio of $0.2 + 0.8 r_{\text{coded}}$, where r_{coded} is the coded-step execution ratio. This distinction explains why token savings and end-to-end speedup need not scale linearly.

J Additional Adaptive Computation and Runtime Analysis

Figure 10 extends Fig. 6 with five High-setting Wan examples. Spatial update maps and per-step counts show how computation varies with both content and observation history. Each token contributes at most one count per sampling step, without duplicating CFG branches or transformer layers. The auxiliary reference-K/V forward pass is timed separately and is not counted as a new sampling-state observation.

Profiling protocol. For each example, we record one complete run per method after a warm-up run on an exclusive A800. Timings use synchronized wall-clock measurements and include text encoding, warm-up, routing, reference preparation, explicit DiT computation, VAE decoding, and video writing. Model loading and offline training are excluded. Semantic-observation and reference-processing spans are subtracted from their enclosing DiT phase before being assigned to separate categories, avoiding double counting.

End-to-end benefit. Table 5 reports $2.42\times$–$2.90\times$ speedups across these examples. Native dense DiT sampling takes approximately 70.4–71.3 s. Our method replaces this with 14.1–14.3 s of dense warm-up and 5.59–10.42 s of subsequent explicit computation. The encoder, learned predictor,

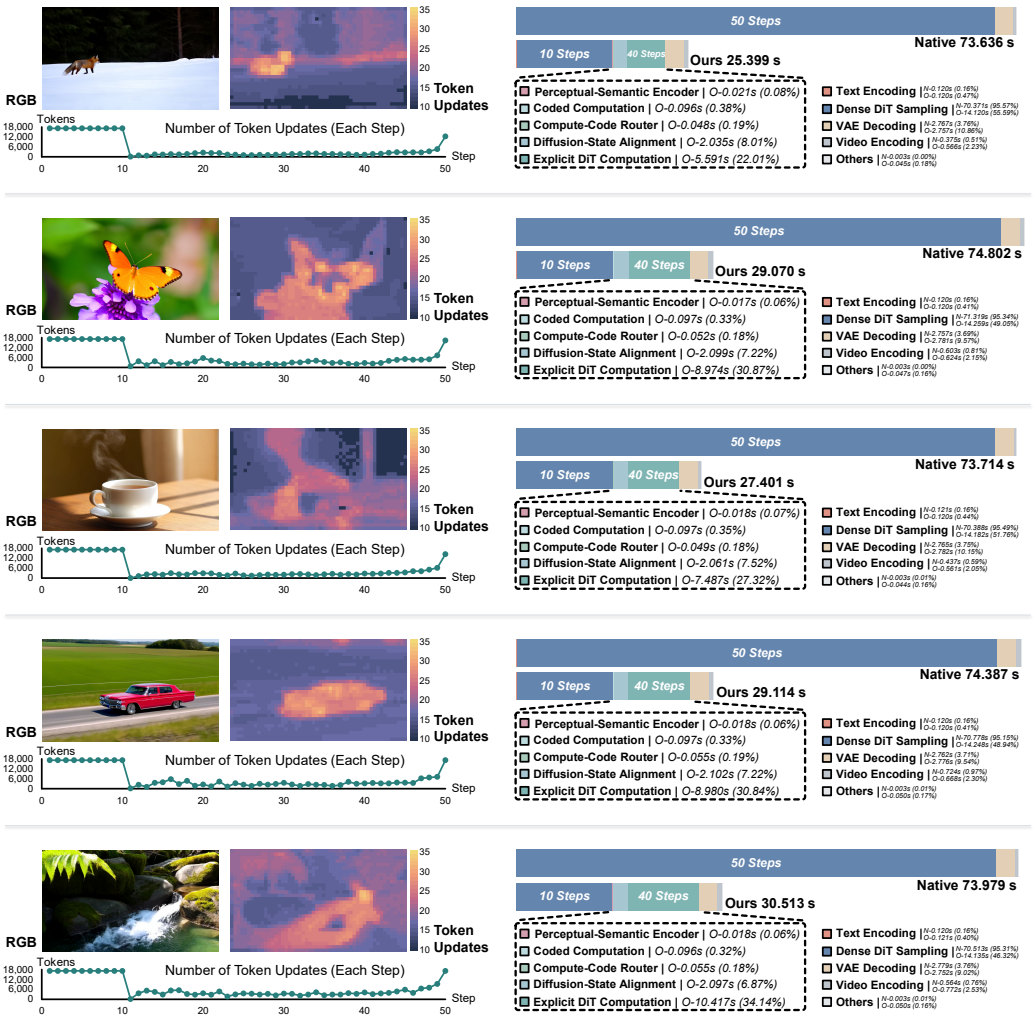

Figure 10: **Additional token-update distributions and runtime breakdowns.** Left: generated frames and cumulative spatial token-update counts. Middle: actual token updates at each denoising step. Right: component-wise runtimes of Native (N) and Ours (O). Rows show a fox, butterfly, steaming cup, car, and stream. Each example uses ten dense warm-up steps and forty coded steps.

Table 5: **Execution ratios and latency for the examples in Fig. 10.** Execution is averaged over the forty coded steps. These are per-scene profiling results, distinct from the dataset averages in Table 1.

Scene	Execution (%)	Native (s)	Ours (s)	Speedup	Reduction
Fox	11.73	73.636	25.399	2.90×	65.5%
Butterfly	20.20	74.802	29.070	2.57×	61.1%
Steaming cup	16.49	73.714	27.401	2.69×	62.8%
Car	20.16	74.387	29.114	2.56×	60.9%
Stream	24.28	73.979	30.513	2.42×	58.8%

and router together cost only 0.163–0.170 s. Including reference construction and alignment brings this overhead to 2.20–2.27 s, well below the saved DiT time. The alignment category includes the additional reference DiT forward pass.

Content-dependent cost. Most runtime variation comes from explicit DiT execution: 5.59 s for the fox versus 10.42 s for the stream. Encoder, predictor, and reference costs vary little. Dense warm-up and about 2.8 s of VAE decoding limit further end-to-end gains from reducing coded-step execution.

K Additional Background Completion and Replacement Results

Figure 11 extends Fig. 7 with twelve examples using the interface in Section D. Animals, a cyclist, and vehicles on land, water, and in the air cover different foreground shapes and scene geometry. Across road, grassland, water, and sky backgrounds, the examples retain recognizable subject outlines and placement while changing the surroundings. External segmentation, completion, and background generation are timed separately from the acceleration benchmark.



Figure 11: **Additional background completion and replacement results.** Left: accelerated generation with the original text conditions. Middle: completed backgrounds. Right: replacement results with the corresponding scene descriptions. All examples use the background-editing interface described in Section D.