REAL: ACCELERATING FLOW MATCHING THROUGH SEGMENT ADVANCEMENT WITH SHARED LOOKAHEAD

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow-matching models generate high-quality images and videos, but repeated neural network evaluations make sampling expensive. Skipping evaluations reduces this cost by extending an available velocity estimate over a longer span. However, local velocity agreement alone does not determine a suitable span, and checking each candidate endpoint adds costly model calls. We introduce ReAL, a training-free sampler that selects how far to advance using one shared lookahead. Our key insight is that the discrepancy between uncorrected and lookahead-corrected endpoint proposals can be computed directly from the observed velocity mismatch and the candidate span beyond the lookahead. This relation provides a span-dependent selection criterion without additional endpoint evaluations. The same lookahead selects the longest passing candidate span, corrects the accepted update, and supplies its velocity as the next starting estimate. After initialization, each regular iteration requires only one fresh evaluation. ReAL uses the pretrained velocity output and original noise schedule, with no additional training or access to internal features. Experiments cover four image-generation backbones, video generation, and image editing. ReAL achieves 4.91× measured speedup on FLUX.1-dev while retaining 97.0% of dense mean ImageReward. On HunyuanVideo, it achieves a 5.49× speedup while maintaining a VBench score close to that of dense sampling.

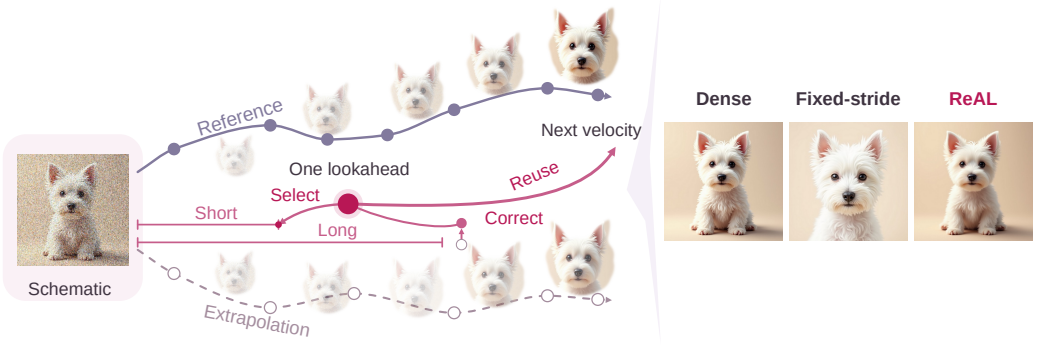

Figure 1: **One lookahead for selection, correction, and reuse.** Left: a schematic of reference sampling, extrapolation, and the three roles of shared lookahead in ReAL. Right: FLUX.1-dev outputs with the same prompt and initial noise, using dense sampling (50 evaluations), fixed-stride sampling (13), and ReAL (9).

1 INTRODUCTION

Flow matching underlies many modern visual generators (Lipman et al., 2023; Esser et al., 2024; Black Forest Labs, 2024). Sampling follows a learned velocity field from noise to data, but each scheduled update invokes a large neural network. Training-free acceleration aims to reduce these velocity evaluations without retraining the model or replacing its backbone.

Existing accelerators pursue this goal by reusing features or predictions, or by advancing across scheduled states with fewer full evaluations through faster solvers, adaptive schedules, and trajectory

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.

forecasts (Lu et al., 2022; Zhao et al., 2023; Liu et al., 2025a;b; Ye et al., 2024). These approaches expose a common tradeoff. Removing a call saves an expensive evaluation, but it also asks an existing velocity estimate to govern more of the trajectory before the field is observed again. A short omission may remain consistent with the current motion, while a longer one can amplify the effect of the same local mismatch. The useful span can also change across sampling and across inputs. Fixed schedules cannot respond to this variation. Endpoint-verification methods can adapt the span, but evaluating a separate endpoint for every candidate returns part of the saved cost (Bajpai et al., 2026). The central question is therefore not only which updates can be omitted, but how far one evaluation can carry sampling from the current state.

Answering this question requires relating *local agreement* to the *advancement span*. A fresh velocity at a nearby lookahead measures disagreement with the available estimate. This observation, however, is local. It compares the available, possibly carried velocity with one fresh lookahead observation. It does not measure the field at later candidate endpoints. Its meaning therefore depends on how far the sampler proposes to advance. The same mismatch yields a larger difference between the raw and corrected endpoint proposals over a longer residual span, so evidence sufficient for a short advance may not justify a longer one. An adaptive controller must judge candidates of different lengths from this shared evidence without spending another evaluation on each alternative. Otherwise, decision overhead can erase the intended saving, while an aggressive choice can rely on evidence that does not support its span. The challenge is to make this span-aware decision from one observation under a fixed evaluation budget. Figure 1 illustrates this tension.

We introduce **ReAL**, a training-free sampler built around *shared lookahead*. ReAL interprets one nearby velocity observation together with each proposed span. The same observation selects the span, corrects the endpoint, and becomes the carried velocity estimate for the next regular iteration. Testing more candidates within that iteration does not require another model evaluation. ReAL uses only the pretrained velocity output and original noise schedule, with no task-specific training or access to intermediate features.

We evaluate ReAL on four image-generation backbones, HunyuanVideo, and Step1X-Edit (Kong et al., 2024; Liu et al., 2025d). It reaches $4.91\times$ measured speedup on FLUX.1-dev with 97.0% of dense mean ImageReward. On HunyuanVideo, it achieves a $5.49\times$ speedup with a 0.54-point drop in VBench score relative to dense sampling. Controlled perturbations produce larger downstream effects at earlier indices, while SD3-Medium traces show shorter early ReAL advances.

Our contributions are:

- We derive a span-scaled proposal-correction discrepancy that makes the acceptance score depend on how far sampling would advance.
- We introduce a shared-lookahead sampler in which one fresh observation selects the span, corrects the accepted endpoint, and supplies the next starting velocity without separate endpoint queries.
- We evaluate image generation, video generation, and image editing, and use ablations and controlled studies to examine selection, correction, reuse, and the placement of evaluations.

2 Related Work

2.1 Flow Models and Fast Sampling

Visual generators span diffusion, score-based, latent diffusion, and transformer models (Ho et al., 2020; Song et al., 2021b; Rombach et al., 2022; Karras et al., 2022; Peebles & Xie, 2023). Flow matching and rectified flow use continuous velocity fields (Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023; Albergo et al., 2025), with work improving paths, couplings, or architectures (Pooladian et al., 2023; Ma et al., 2024a; Esser et al., 2024). Fixed-model accelerators use implicit updates, higher-order integration, predictor-corrector formulas, and optimized schedules (Song et al., 2021a; Liu et al., 2022; Lu et al., 2022; 2025; Zhang & Chen, 2023; Zhao et al., 2023; Xu et al., 2023b; Xue et al., 2024; Sabour et al., 2024). Others train few-step generators through distillation, consistency, trajectory, or adversarial objectives (Salimans & Ho, 2022; Song et al., 2023; Luo et al., 2023; Meng et al., 2023; Kim et al., 2024; Liu et al., 2024; Sauer et al., 2024). ReAL preserves the field and schedule, adapting where evaluations occur.

2.2 Reuse and Adaptive Computation

Reuse methods cache slowly changing features (Ma et al., 2024c;b). TeaCache and AdaCache track change, while TaylorSeer and SpeCa forecast activations (Liu et al., 2025a; Kahatapitiya et al., 2025; Liu et al., 2025b;c). Adaptive methods use spectral change, path cost, velocity structure, future prediction, spatial selection, or latent stability (Chung et al., 2026; Bu et al., 2026; Cui et al., 2026; Han et al., 2026; Tan et al., 2026; Wang et al., 2026; Sun et al., 2026; Ye et al., 2024; Jiang et al., 2025). SeaCache filters intermediate features in the frequency domain, while VDE estimates components of the velocity change (Chung et al., 2026; Tan et al., 2026). Spatial selection, prediction caching, attention fast paths, early exit, and token merging provide complementary savings within or between model calls (Yin et al., 2026c;a; Moon et al., 2024; Bolya & Hoffman, 2023). ReAL instead uses complete model outputs, requires no feature cache or learned predictor, and carries the fresh lookahead velocity across segment boundaries.

2.3 Lookahead across Visual Tasks

Parallel samplers update multiple denoising states together (Shih et al., 2023; Chen et al., 2024b;c), while speculative decoding drafts and verifies autoregressive tokens (Leviathan et al., 2023). Diffusion methods verify transitions or forecast flow trajectories (De Bortoli et al., 2025; Bajpai et al., 2026), and FlowTurbo trains a velocity refiner (Zhao et al., 2024). FlowCast drafts a constant-velocity trajectory and verifies drafted states with parallel model evaluations. ReAL queries only the next-index lookahead, uses no candidate-endpoint query, and shares that output across span selection, endpoint correction, and velocity reuse. We evaluate this training-free controller on image, video, and editing. Video generators include Video Diffusion Models, VideoCrafter2, and HunyuanVideo (Ho et al., 2022; Chen et al., 2024a; Kong et al., 2024), with VBench measuring visual and temporal quality (Huang et al., 2024). Image editing covers stochastic transformation, instruction following, annotated evaluation, and inversion-free flow editing (Meng et al., 2022; Brooks et al., 2023; Zhang et al., 2023; Kulikov et al., 2025).

3 Methodology

ReAL uses one lookahead to select and correct a segment, then carries its velocity forward. Figure 2 summarizes the controller.

3.1 Solver State and One-Step Lookahead

Let x_i denote the sampler state at index i, where σ_i is its noise level and $\sigma_0 > \cdots > \sigma_T$. Here T is the final index, $\Delta\sigma_i = \sigma_{i+1} - \sigma_i$, and $K \in \mathbb{N}$, $K \geq 3$, is the maximum candidate length in schedule intervals. Let $f(x, \sigma)$ denote the pretrained velocity field, or the guided field under classifier-free guidance. The dense reference applies $x_{i+1} = x_i + \Delta\sigma_i f(x_i, \sigma_i)$. ReAL retains this model and schedule but may skip intermediate indices.

At index i, u_i is the starting velocity, normally carried from the previous iteration. Without a carry, ReAL evaluates $u_i = f(x_i, \sigma_i)$. It then forms the lookahead:

$$z_i = x_i + \Delta\sigma_i u_i, \qquad w_i = f(z_i, \sigma_{i+1}), \tag{1}$$

where z_i is the one-step lookahead state and w_i is its fresh velocity. Equation (1) uses the available starting velocity and one fresh model query. All candidates share (u_i, w_i).

3.2 A Corrected Segment and Its Verification Signals

For a candidate of $k \geq 3$ intervals with $i + k \leq T$, freezing u_i gives the raw endpoint $\hat{x}_{i+k|i} = x_i + (\sigma_{i+k} - \sigma_i)u_i$. ReAL instead uses:

$$\tilde{x}_{i+k|i} = z_i + (\sigma_{i+k} - \sigma_{i+1})w_i, \tag{2}$$

where the first interval uses u_i, the remaining span uses w_i, and $i+k \mid i$ records the target and starting indices. Equation (2) therefore corrects the remainder of the proposed segment without another query. The raw and corrected endpoints differ by:

$$\hat{x}_{i+k|i} - \tilde{x}_{i+k|i} = (\sigma_{i+k} - \sigma_{i+1})(u_i - w_i). \tag{3}$$

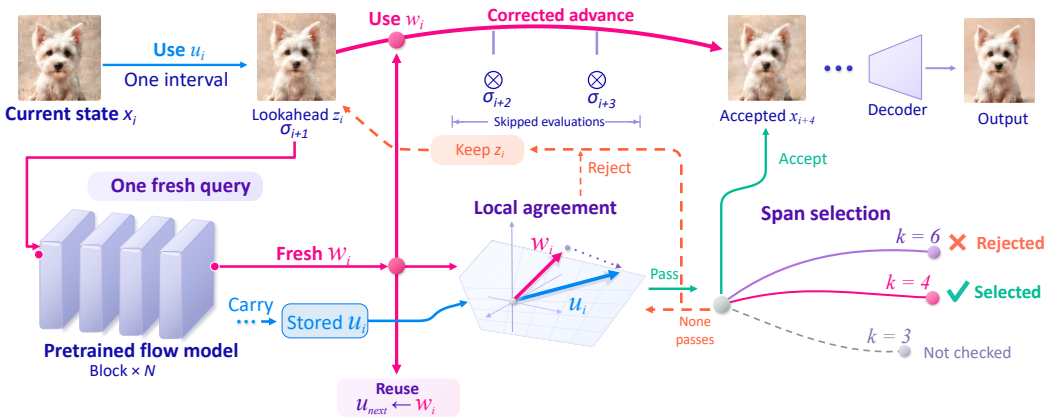

Figure 2: **One lookahead serves three roles:** segment selection, endpoint correction, and velocity reuse. The blue carry supplies stored u_i, and the fresh query at z_i produces shared w_i. State images are illustrative. The regular iteration is shown here, with initialization and tail handling specified in Algorithm 1.

Thus the observed velocity mismatch exactly determines the difference between the raw and corrected endpoints for any stored velocity. Appendix A distinguishes it from dense-trajectory error.

For a vector $v \in \mathbb{R}^d$ with the flattened dimension of the sampler state, define $\|v\|_{\mathrm{rms}} = (d^{-1}\sum_{\ell=1}^{d} v_\ell^2)^{1/2}$, where ℓ indexes coordinates. Let $h_i = |\Delta\sigma_i|$ be the local interval and $s_{i,k} = |\sigma_{i+k} - \sigma_{i+1}|$ be the residual span. ReAL uses:

$$r_i^{\mathrm{loc}} = \frac{\|u_i - w_i\|_{\mathrm{rms}}}{\|u_i\|_{\mathrm{rms}} + \epsilon_v}, \qquad r_{i,k}^{\mathrm{seg}} = \frac{s_{i,k}}{h_i + \epsilon_\sigma} r_i^{\mathrm{loc}}, \tag{4}$$

where r_i^{loc} is the local mismatch, $r_{i,k}^{\mathrm{seg}}$ scores candidate k, and $\epsilon_v = \epsilon_\sigma = 10^{-8}$ are stability constants. The local residual compares velocity direction and magnitude together, as shown in Appendix Eq. (8). By Eq. (3), the segment score normalizes the RMS difference between the raw and corrected endpoints by $(h_i + \epsilon_\sigma)(\|u_i\|_{\mathrm{rms}} + \epsilon_v)$. The first-interval reference keeps this denominator fixed across candidates. Longer residual spans therefore receive larger scores for the same nonzero mismatch. Normalizing by the residual span would instead cancel this length dependence.

Using the actual differences in σ, rather than index differences, accommodates nonuniform schedules. The nonnegative q_{loc} and q_{seg} are the acceptance thresholds for local and segment mismatch. Both tests use the same guided field and determine the advance through the backoff procedure below.

3.3 Hierarchical Backoff, Fallback, and Carry

Set $K_a = \min(K, T - i)$, where K_a is the remaining-interval cap. For $K_a \geq 3$, ReAL forms $\mathcal{C}_i$ from K_a, $\max(3, \lfloor 3K_a/4 \rfloor)$, $\max(3, \lfloor K_a/2 \rfloor)$, and $\max(3, \lfloor K_a/4 \rfloor)$, removes repeated lengths, and tests the result in descending order. Appendix Eq. (5) gives the set notation. For example, $K_a = 6$ gives $\mathcal{C}_i = \{6, 4, 3\}$. The cap sets the longest proposal, while the thresholds determine the actual advance. Each backoff step rescales the shared mismatch for a shorter residual span. Once a candidate passes, ReAL constructs its corrected endpoint using the same w_i.

If $r_i^{\mathrm{loc}} > q_{\mathrm{loc}}$, ReAL falls back to $x_{i+1} = z_i$. Otherwise, it tests $\mathcal{C}_i$ in descending order. The first candidate with $r_{i,k}^{\mathrm{seg}} \leq q_{\mathrm{seg}}$ is accepted, and ReAL sets $x_{i+k} = \tilde{x}_{i+k|i}$. If no candidate passes, it again uses $x_{i+1} = z_i$. All candidate scores use the same two velocities, so backoff requires no additional model evaluation.

After either outcome, ReAL stores $u_{\mathrm{next}} \leftarrow w_i$, where u_{next} is the next starting velocity. Fallback stores the velocity evaluated at the accepted state $z_i = x_{i+1}$, while a longer segment carries it from the earlier lookahead. The first interval matches dense Euler when $u_i = f(x_i, \sigma_i)$. With fewer than three intervals remaining, ReAL consumes any carry once and evaluates later tail states. Each regular iteration after initialization therefore uses one fresh query. Algorithm 1 includes initialization and all tail evaluations.

Table 1: **FLUX.1-dev text-to-image comparison.** Results use 200 DrawBench prompts, five seeds, and 1024^2 resolution. Superscripts 1, 2, and 3 denote progressively faster operating points. Bold marks the best accelerated quality value within each tier. Dashes denote unavailable dense-reference similarity fields.

FLUX.1-dev	Latency (s)↓	Efficiency Speedup↑	FLOPs (10^{12})↓	FLOPs ratio↑	Mem (GB)↓	Image Reward↑	CLIP Score↑	PSNR↑	SSIM↑	LPIPS↓
50 steps	11.38	1.00×	2916.7	1.00×	31.92	0.9907	18.73	-	-	-
40% steps	4.50	2.53×	1126.1	2.59×	31.92	0.9468	18.31	16.88	0.7378	0.3216
30% steps	3.49	3.26×	875.89	3.33×	31.92	0.9381	17.92	15.03	0.6523	0.4279
22% steps	2.60	4.38×	655.4	4.45×	31.92	0.8917	17.36	13.79	0.6032	0.4683
TeaCache[1]	3.91	2.91×	982.1	2.97×	31.97	0.9724	17.85	16.09	0.6679	0.3964
TaylorSeer[1]	3.68	3.09×	786.2	3.71×	37.89	0.9768	18.60	17.83	0.7268	0.2935
DiCache[1]	4.14	2.75×	1027.0	2.84×	33.57	0.9737	17.61	18.62	0.7759	0.3068
SpeCa[1]	3.25	3.50×	645.3	4.52×	36.01	0.9979	18.60	19.39	0.8123	**0.1489**
DPCache[1]	3.99	2.85×	962.6	3.03×	32.08	0.9973	18.63	21.79	0.8093	0.1535
Spectrum[1]	3.84	2.96×	934.8	3.12×	36.62	0.9933	**18.73**	**23.70**	0.8177	0.1811
FlowCast[1]	4.84	2.35×	1178.3	2.48×	31.94	0.9849	18.54	20.82	0.8071	0.1904
ReAL[1]	3.47	3.28×	782.0	3.73×	31.93	**0.9981**	18.69	23.30	**0.8247**	0.2060
TeaCache[2]	3.13	3.63×	790.4	3.69×	31.97	0.9676	17.74	15.12	0.6249	0.4725
TaylorSeer[2]	2.80	4.06×	630.0	4.63×	37.89	0.9644	18.40	15.80	0.6518	0.4009
DiCache[2]	3.72	3.06×	911.5	3.20×	33.64	0.9317	17.50	15.85	**0.8328**	0.1846
SpeCa[2]	2.75	4.14×	535.2	5.45×	36.01	0.9746	18.38	18.73	0.7993	0.2103
DPCache[2]	3.26	3.49×	773.7	3.77×	32.08	0.9812	18.47	20.12	0.8093	**0.1835**
Spectrum[2]	3.40	3.35×	817.0	3.57×	33.57	0.9792	18.60	**22.13**	0.8052	0.2124
FlowCast[2]	3.48	3.27×	816.7	3.57×	31.94	0.9716	18.38	18.74	0.7688	0.2579
ReAL[2]	2.71	4.20×	601.4	4.85×	31.93	**0.9864**	**18.62**	21.14	0.8155	0.2295
TeaCache[3]	2.83	4.02×	711.4	4.10×	31.97	0.8038	17.67	14.84	0.6109	0.5013
TaylorSeer[3]	2.36	4.83×	481.3	6.06×	37.89	0.7809	18.35	14.03	0.5843	0.5134
DiCache[3]	3.50	3.25×	852.8	3.42×	33.57	0.8503	17.39	14.33	0.7703	0.2975
SpeCa[3]	2.45	4.65×	457.9	6.37×	36.01	0.9412	17.94	16.89	0.7421	0.3218
DPCache[3]	2.36	4.82×	539.1	5.41×	32.08	0.9590	18.39	18.99	0.7189	0.2982
Spectrum[3]	2.46	4.62×	579.9	5.03×	33.64	0.9602	18.47	**20.32**	0.7976	0.2649
FlowCast[3]	2.77	4.11×	641.7	4.55×	31.94	0.9473	18.18	17.06	0.7186	0.3374
ReAL[3]	2.32	4.91×	554.5	5.26×	31.93	**0.9611**	**18.55**	19.96	**0.8045**	**0.2551**

4 Experiments and Results

We evaluate quality and latency across image generation, video generation, and editing, isolate the three controller components, and examine backbone transfer and evaluation allocation under changes in cap, schedule, and prompts.

4.1 Experimental Settings

All experiments were conducted on a single NVIDIA H100 GPU. Every latency ratio compares a method with its dense reference under the same precision and timing protocol. We test four text-to-image backbones (Black Forest Labs, 2024; Esser et al., 2024; Zhuo et al., 2024) on DrawBench (Saharia et al., 2022), with an additional SD3-Medium study on a 150-prompt PartiPrompts subset (Yu et al., 2022). We evaluate HunyuanVideo on 946 VBench prompts (Kong et al., 2024; Huang et al., 2024) and Step1X-Edit on 606 GEdit-Bench examples (Liu et al., 2025d).

Metrics, Baselines, and Protocol. ImageReward (IR), CLIPScore, the VBench aggregate, and the three GEdit-Bench scores measure task quality. Peak signal-to-noise ratio (PSNR), structural similarity (SSIM), and learned perceptual image patch similarity (LPIPS) measure agreement with matched dense outputs (Xu et al., 2023a; Hessel et al., 2021; Wang et al., 2004; Zhang et al., 2018). Efficiency uses floating-point operations (FLOPs), number of function evaluations (NFE), and measured end-to-end latency under matched conditions. FLOPs and NFE describe arithmetic and model-call costs. Latency also includes encoding, decoding, tensor operations, and execution overhead. We compare twelve acceleration baselines plus fixed-stride and dense-head controls (Liu

Table 2: **HunyuanVideo on VBench.** Results use 946 prompts, five seeds, 544×960, and 65 frames. Symbols † and ‡ denote medium and high acceleration. Step reduction is the control. Bold marks the best accelerated result within each tier. Latency is median event time, speedup uses the matched dense run, and Mem is peak allocation. Appendix B gives the protocol.

Method	Efficiency						Generation quality			
	Latency (s)↓	Speedup↑	FLOPs (10^{12})↓	FLOPs ratio↑	NFE↓	Mem (GB)↓	VBench↑	PSNR↑	SSIM↑	LPIPS↓
Dense 50 steps	298.41	1.00×	23750.0	1.00	50.0	60.44	83.27	N/A	N/A	N/A
22% steps	63.47	4.70×	4979.0	4.77	10.7	60.44	80.36	19.49	0.7002	0.4228
TeaCache†	84.12	3.55×	5549.1	4.28	11.5	60.56	81.39	21.94	0.7339	0.2077
TaylorSeer†	79.73	3.74×	5031.8	4.72	10.7	93.09	82.73	20.21	0.7549	0.2557
DiCache†	98.81	3.02×	6805.2	3.49	14.1	60.48	82.87	25.96	0.8206	0.2020
SpeCa†	80.03	3.73×	4827.2	4.92	10.0	92.44	82.53	24.80	0.7624	0.2462
DPCache†	81.98	3.64×	5952.4	3.99	12.3	62.10	82.96	25.97	0.8131	0.2031
Spectrum†	92.84	3.21×	6418.9	3.70	13.7	70.91	**83.14**	26.16	0.8166	0.1982
FlowCast†	89.68	3.33×	6650.0	3.57	14.0	60.73	82.58	25.40	0.8025	0.2161
SeaCache†	81.12	3.68×	5730.2	4.14	11.8	62.83	82.88	26.13	0.8174	0.1963
VDE†	86.71	3.44×	5826.1	4.08	12.6	66.37	82.64	25.54	0.8081	0.2113
ZEUS†	76.16	3.92×	5392.8	4.40	11.3	64.92	82.47	25.09	0.7992	0.2241
AdaptiveDiffusion†	94.38	3.16×	6248.8	3.80	13.4	61.34	82.21	24.71	0.7923	0.2318
JiT†	82.63	3.61×	5638.8	4.21	12.1	71.62	82.39	25.28	0.8034	0.2187
ReAL†	74.27	**4.02**×	5186.3	4.58	11.2	60.72	83.09	26.44	0.8247	0.1871
TeaCache‡	71.30	4.19×	5000.0	4.75	10.5	62.48	80.42	16.91	0.6666	0.3662
TaylorSeer‡	63.72	4.68×	4159.4	5.71	8.5	92.58	81.43	17.29	0.6958	0.3792
DiCache‡	64.24	4.65×	4541.1	5.23	9.5	62.10	81.38	18.65	0.7375	0.2853
SpeCa‡	62.29	4.79×	4018.6	5.91	8.6	93.91	80.69	16.68	0.6733	0.3487
DPCache‡	70.21	4.25×	4515.2	5.26	9.7	61.14	82.01	19.53	0.7411	0.2956
Spectrum‡	67.36	4.43×	4481.1	5.30	9.2	67.38	82.31	19.60	0.7597	0.2696
FlowCast‡	68.76	4.34×	5272.5	4.50	11.3	62.04	81.72	19.23	0.7266	0.3028
SeaCache‡	61.81	4.83×	4302.1	5.52	9.0	62.11	82.06	19.92	0.7541	0.2806
VDE‡	65.27	4.57×	4474.0	5.31	9.6	65.18	81.71	19.37	0.7443	0.2961
ZEUS‡	60.76	4.91×	4042.4	5.88	8.7	63.74	81.39	18.84	0.7328	0.3117
AdaptiveDiffusion‡	73.19	4.08×	5040.7	4.71	10.4	61.02	80.98	18.31	0.7214	0.3269
JiT‡	64.02	4.66×	4468.9	5.31	9.3	69.83	81.24	19.03	0.7382	0.3034
ReAL‡	54.40	**5.49**×	3944.7	6.02	8.1	60.68	**82.73**	20.58	0.7696	0.2681

et al., 2025a;b; Bu et al., 2026; Liu et al., 2025c; Cui et al., 2026; Han et al., 2026; Bajpai et al., 2026; Chung et al., 2026; Tan et al., 2026; Wang et al., 2026; Ye et al., 2024; Sun et al., 2026). Dense-head uses m exact prefix steps and later fixed span k, written $m{:}k$, while retaining ReAL's correction and carry. Each operating point is fixed across its reported prompts and seeds. Appendices B and C give complete settings and implementations.

4.2 Text-to-Image Generation

Quantitative Comparison. At the fastest of three FLUX tiers, ReAL reaches $4.91\times$ speedup with IR 0.9611 and LPIPS 0.2551 (Table 1). Direct step reduction reaches $4.38\times$ with IR 0.8917 and LPIPS 0.4683. FlowCast reaches $4.11\times$ with IR 0.9473 and LPIPS 0.3374.

4.3 Text-to-Video Generation

On HunyuanVideo, ReAL leads the high-acceleration settings at $5.49\times$ speedup, reducing latency from 298.41 to 54.40 seconds at 8.1 NFE (Table 2). Its VBench score is 82.73, 0.54 points below dense sampling. Direct step reduction reaches $4.70\times$ with 80.36 VBench.

At medium acceleration, ReAL reaches $4.02\times$ with 83.09 VBench, while Spectrum reaches $3.21\times$ with 83.14. At high acceleration, ReAL records 0.7696 SSIM and 0.2681 LPIPS, compared with 0.7266 and 0.3028 for FlowCast. Its dynamic-degree score is 66.94, versus 71.11 for dense sampling and 43.33 for direct step reduction. Appendix B.2 reports all sixteen dimensions.

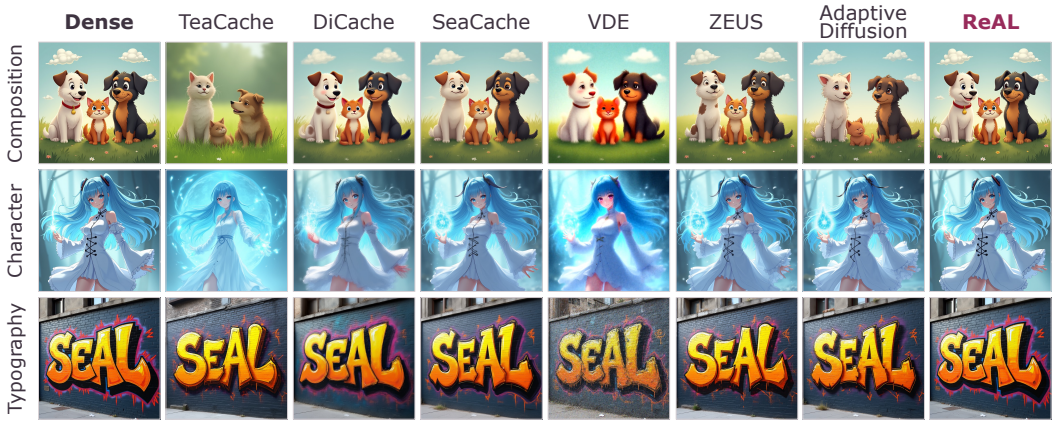

Figure 3: **FLUX.1-dev qualitative comparison.** ReAL retains the requested three-object composition, cyan-haired character, and literal word "SEAL" in these matched examples. The rows use identical prompts, initial noise, and text embeddings across methods. Each method uses one fixed configuration. Appendix F gives the full prompts, settings, and additional comparisons.

Matched Generation Comparisons. Figure 3 compares text-to-image outputs, and Figure 4 shows matched HunyuanVideo frames. Both comparisons use shared prompts and initial noise within each task.

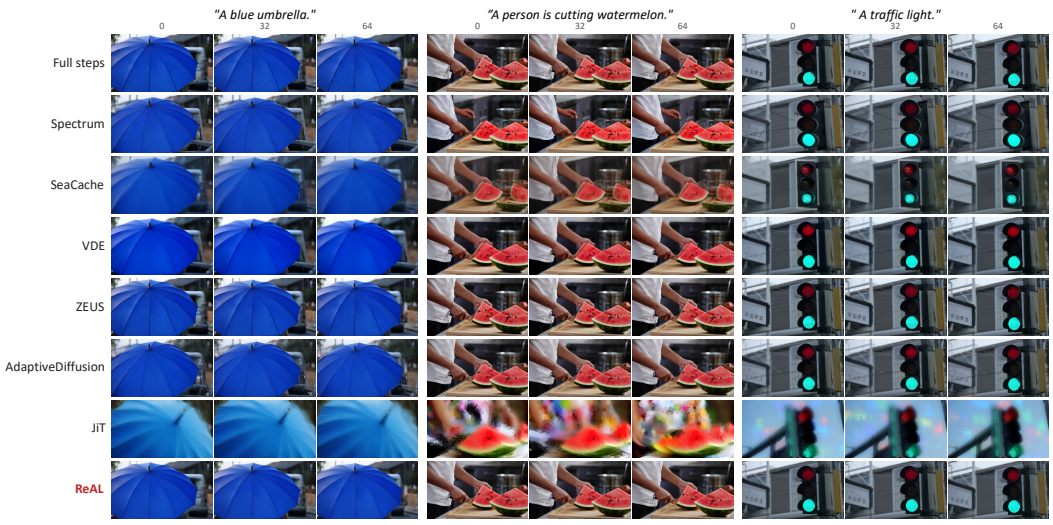

Figure 4: **HunyuanVideo qualitative comparison.** Matched-seed bf16 runs compare ReAL with seven reference methods at frames 0, 32, and 64. Table 2 uses separate fp16 timing runs. Appendix G gives all fifteen methods and settings.

4.4 Image Editing

ReAL has the highest overall-score point estimate in Tiers 1 and 2, reaching 4.24 at $3.86\times$ and 4.21 at $4.58\times$, compared with 4.20 for dense sampling. The available summary does not include seed-level intervals, so small differences such as 4.24 versus 4.20 should not be interpreted as statistically resolved. In Tier 3, ReAL is fastest at $5.22\times$, with a score of 4.11 and latency of 1.311 seconds, versus 6.842 seconds for dense sampling. TaylorSeer scores 4.16 at $4.78\times$. Figure 5 shows all three editing scores, and Table 5 gives the full 22-setting comparison.

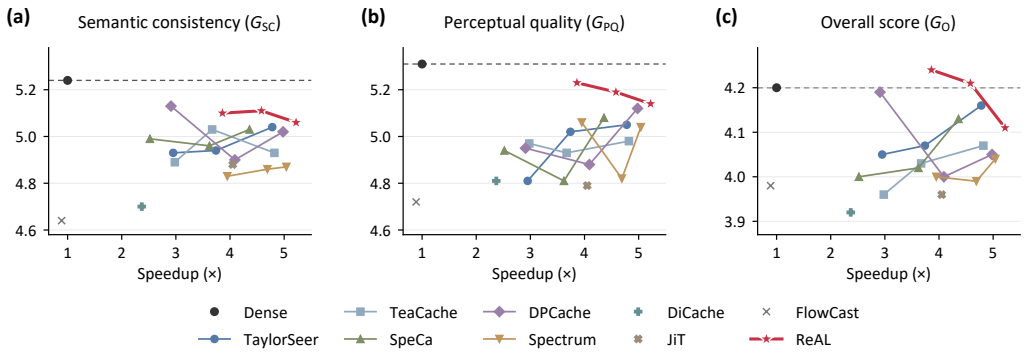

Figure 5: **Editing quality across acceleration settings.** Step1X-Edit on 606 examples with five shared seeds. Lines connect measured settings, and dashed lines mark dense quality. Table 5 reports all 22 settings.

Table 3: **Selection, correction, and cross-segment reuse on FLUX.** Panels (a) and (b) use a separate fallback-only diagnostic protocol and should not be read as replacements for the final current-method comparison in Table 1. Panel (c) is a 100-prompt carry-policy comparison at $q_{\text{seg}} = 1.65$. The appendix reports full settings and statistics.

(a) Which Segments Should Advance?

Verification	Speedup↑	IR↑	NFE↓
None	9.86×	−0.0168	5.0
Local only	7.96×	0.4395	6.0
Span only	4.58×	0.9621	9.8
Both criteria	4.20×	**0.9864**	10.3
Dense verification	0.42×	0.7801	118.4

(b) How Should the Endpoint Update?

Endpoint	Speedup↑	IR↑	NFE↓
Raw proposal	5.23×	0.6492	8.76
Corrected	4.91×	**0.9611**	9.51

(c) Reuse across Long Segments

Carry policy	IR↑	NFE↓	Calls/iter.↓
Only if $k \leq 4$	0.8632	11.06	1.728
All segments	0.8719	**9.48**	**1.062**

4.5 Ablation of Shared-Lookahead Components

Under fallback-only carry, span verification raises IR from -0.0168 without verification to 0.9621. Adding the local gate reaches 0.9864 at $4.20\times$ speedup. The span test scales velocity mismatch by the residual schedule span relative to the first interval. Dense verification requires 118.4 NFE and reaches only $0.42\times$ speedup.

Endpoint correction raises IR from 0.6492 to 0.9611 at the fastest tested setting, while NFE changes from 8.76 to 9.51. The correction reuses the selection query. At $q_{\text{seg}} = 1.65$, carrying lookahead across all accepted segments reduces NFE from 11.06 to 9.48, or 14.3%, and calls per iteration from 1.728 to 1.062. IR changes from 0.8632 to 0.8719.

These controls show distinct benefits from span selection, endpoint correction, and velocity carry. Local verification alone ignores span length, while dense endpoint checks add model calls. Sharing w_i supports all three decisions without a separate query for correction or reuse.

4.6 Transfer across Backbones and Sampling Settings

Figure 6 compares ImageReward and NFE on DrawBench-100 for three additional backbones, each with its own thresholds. Further studies vary the segment cap, schedule, and prompts to examine evaluation allocation.

Backbone Transfer. On SD3-Medium, ReAL reaches 0.594 IR at 12.43 NFE, compared with 0.591 at 13.59 NFE for AdaptiveDiffusion. On SD3.5-Large, ReAL and TeaCache both round to 0.794 IR at

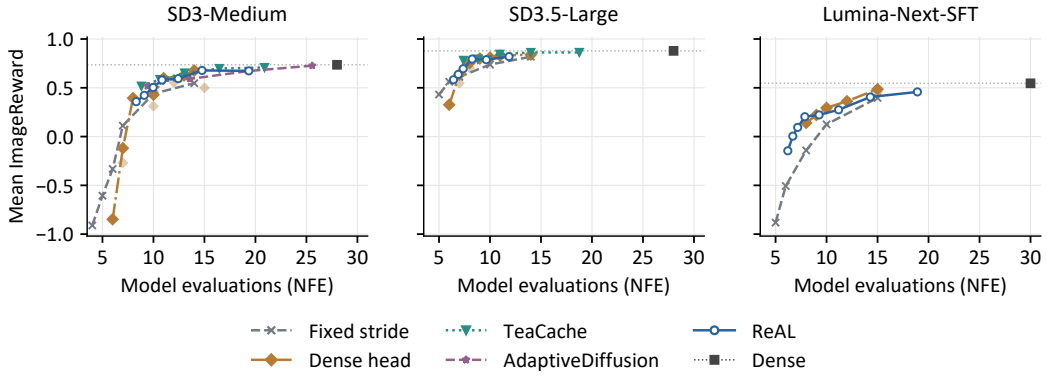

Figure 6: **Quality versus compute on three additional backbones.** On DrawBench-100, ReAL uses $K = 6$ and a threshold sweep per backbone. Points are measured settings, and lines guide the eye. The dense-head line connects its best measured trade-offs, with other settings shown faintly. TeaCache uses output-delta reuse on the SD3 backbones, and AdaptiveDiffusion uses the flow adaptation on SD3-Medium. Numerical comparisons appear in Appendix C.

8.25 and 8.80 NFE, respectively. The dense-head rule is stronger on Lumina at intermediate budgets. These sweeps evaluate the controller across backbones with settings chosen for each workload.

Segment-Cap Changes. With $q_{\text{seg}} = 2.6$ fixed, increasing K from 12 to 24 keeps ReAL between 7.65 and 7.83 NFE, with IR between 0.853 and 0.904. The fixed rule instead reduces NFE from 8 to 6 while IR falls from 0.893 to 0.283. At $K = 24$, ReAL's paired IR advantage is 0.621, with a 95% bootstrap CI of [0.481, 0.765], using 1.83 more evaluations. Table 10 reports both threshold sweeps.

Schedule and Prompt Changes. The 200-prompt SD3-Medium study in Appendix Table 12 compares 28-step and 15-step schedules. At 15 steps, ReAL reaches 0.1979 IR at 6.245 NFE, versus 0.0470 at 6 NFE for rule 3:6. On 150 PartiPrompts at $q_{\text{seg}} = 1.65$, ReAL uses 11.52 NFE for the 98 prompts outside the Basic and Simple Detail categories, versus 10.42 for the other 52, a 10.5% increase without category labels as inputs. Appendix Figure 8 plots these allocation changes.

Allocation and Error Propagation. Early perturbations have larger final effects across three backbones. On SD3-Medium, mean terminal drift divided by mean summed local discrepancy is 3.40 for uniform stride two, 1.04 for a dense head, and 1.21 for ReAL. Appendix E defines this diagnostic and plots ReAL's shorter early advances in Figure 10.

5 Conclusion

ReAL frames training-free flow-matching acceleration as deciding how far sampling can advance from local evidence. One lookahead compares the stored and fresh velocities, selects a span, corrects the endpoint, and supplies the next starting velocity. ReAL therefore leaves the pretrained model and noise schedule unchanged and avoids separate candidate-endpoint queries.

Across the evaluated image, video, and editing systems, ReAL reduces latency at the reported quality levels. It reaches $4.91\times$ speedup on FLUX.1-dev with 97.0% of dense mean ImageReward. On HunyuanVideo, it achieves a $5.49\times$ speedup with a 0.54-point drop in VBench score relative to dense sampling. The ablations and allocation studies show how verification, correction, carry, and evaluation placement affect measured quality and cost. Local velocity disagreement can guide where computation is useful, but it is not a calibrated estimate of final error.

New deployments require model-specific calibration because later field changes and long segments can weaken local evidence and velocity carry. Thresholds should be chosen using task quality and measured latency on the target model and schedule. Controller traces can then show how often fallback occurs and which spans are selected.

AI USE STATEMENT

AI tools assisted with manuscript drafting and editing, literature checks.

ETHICS STATEMENT

ReAL changes inference of existing generative models and inherits their risks of biased, misleading, or harmful content. The reported perceptual and alignment metrics do not assess these risks. Faster inference should not be interpreted as improved factuality or safety.

REPRODUCIBILITY STATEMENT

The appendix specifies the controller, candidate construction, carry, tail handling, evaluation settings, and statistical comparisons. The supplementary material includes a model-independent implementation of Algorithm 1, a CPU example, an end-to-end FLUX.1-dev single-image entry point, and tests of selection, correction, fallback, carry, tail handling, and NFE counting. The source package also contains frozen result summaries, operating points, provenance records, and scripts that audit the reported comparisons, regenerate the numerical figures, and rebuild the paper. Pretrained model and evaluator weights are not redistributed. Full task reruns require the documented external weights, benchmark inputs, environments, seeds, and configurations. Raw generations and per-sample traces are stored separately because of their size.

REFERENCES

Michael Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025.

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *International Conference on Learning Representations (ICLR)*, 2023.

Divya Jyoti Bajpai, Shubham Agarwal, Apoorv Saxena, Kuldeep Sharad Kulkarni, Subrata Mitra, and Manjesh Kumar Hanawal. FlowCast: Trajectory forecasting for scalable zero-cost speculative flow matching. In *International Conference on Learning Representations (ICLR)*, volume 2026, pp. 91560–91580, 2026.

Black Forest Labs. FLUX.1 [dev]. Official model card, 2024.

Daniel Bolya and Judy Hoffman. Token merging for fast Stable Diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*, pp. 4599–4603, 2023.

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. InstructPix2Pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18392–18402, 2023.

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 (ICLR)*, volume 2026, pp. 73778–73803, 2026.

Haoxin Chen, Yong Zhang, Xiaodong Cun, Menghan Xia, Xintao Wang, Chao Weng, and Ying Shan. VideoCrafter2: Overcoming data limitations for high-quality video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7310–7320, 2024a.

Haoxuan Chen, Yinuo Ren, Lexing Ying, and Grant Rotskoff. Accelerating diffusion models with parallel sampling: Inference at sub-linear time complexity. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 133661–133709, 2024b.

Zigeng Chen, Xinyin Ma, Gongfan Fang, Zhenxiong Tan, and Xinchao Wang. AsyncDiff: Parallelizing diffusion models by asynchronous denoising. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 95170–95197, 2024c.

Jiwoo Chung, Sangeek Hyun, MinKyu Lee, Byeongju Han, Geonho Cha, Dongyoon Wee, Youngjun Hong, and Jae-Pil Heo. SeaCache: Spectral-evolution-aware cache for accelerating diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14283–14294, 2026.

Bowen Cui, Yuanbin Wang, Huajiang Xu, Biaolong Chen, Aixi Zhang, Hao Jiang, Zhengzheng Jin, Xu Liu, and Pipei Huang. Denoising as path planning: Training-free acceleration of diffusion models with DPCache. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 43632–43642, 2026.

Valentin De Bortoli, Alexandre Galashov, Arthur Gretton, and Arnaud Doucet. Accelerated diffusion models via speculative sampling. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267, pp. 12590–12631, 2025.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235, pp. 12606–12633, 2024.

Jiaqi Han, Juntong Shi, Puheng Li, Haotian Ye, Qiushan Guo, and Stefano Ermon. Adaptive spectral feature forecasting for diffusion sampling acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 43320–43330, 2026.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A reference-free evaluation metric for image captioning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7514–7528, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 6840–6851, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David Fleet. Video diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 8633–8646, 2022.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21807–21818, 2024.

Ting Jiang, Yixiao Wang, Hancheng Ye, Zishan Shao, Jingwei Sun, Jingyang Zhang, Zekai Chen, Jianyi Zhang, Yiran Chen, and Hai Li. SADA: Stability-guided adaptive diffusion acceleration. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267, pp. 27649–27669, 2025.

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 (ICCV)*, pp. 15240–15252, 2025.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 26565–26577, 2022.

Dongjun Kim, Chieh-Hsin Lai, WeiHsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In *International Conference on Learning Representations (ICLR)*, volume 2024, pp. 44493–44525, 2024.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-Pic: An open dataset of user preferences for text-to-image generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 36652–36663, 2023.

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, 2024. arXiv preprint arXiv:2412.03603.

Vladimir Kulikov, Matan Kleiner, Inbar Huberman-Spiegelglas, and Tomer Michaeli. FlowEdit: Inversion-free text-based editing using pre-trained flow models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19721–19730, 2025.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202, pp. 19274–19286, 2023.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

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

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with TaylorSeers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15853–15863, 2025b.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Fei Ren, Shaobo Wang, Kaixin Li, and Linfeng Zhang. SpeCa: Accelerating diffusion transformers with speculative feature caching. In *Proceedings of the 33rd ACM International Conference on Multimedia (ACM MM)*, pp. 10024–10033, 2025c.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations (ICLR)*, 2022.

Shiyu Liu, Yucheng Han, Peng Xing, Fukun Yin, Rui Wang, Wei Cheng, Jiaqi Liao, Yingming Wang, Honghao Fu, Chunrui Han, Guopeng Li, Yuang Peng, Quan Sun, Jingwei Wu, Yan Cai, Zheng Ge, Ranchen Ming, Lei Xia, Xianfang Zeng, Yibo Zhu, Binxing Jiao, Xiangyu Zhang, Gang Yu, and Daxin Jiang. Step1X-Edit: A practical framework for general image editing, 2025d. arXiv preprint arXiv:2504.17761.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023.

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, and Qiang Liu. InstaFlow: One step is enough for high-quality diffusion-based text-to-image generation. In *International Conference on Learning Representations (ICLR)*, volume 2024, pp. 17860–17889, 2024.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 5775–5787, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 22 (4):730–751, 2025.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference, 2023. arXiv preprint arXiv:2310.04378.

Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. SiT: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision (ECCV)*, volume 15135 of *Lecture Notes in Computer Science*, pp. 23–40, 2024a.

Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-Cache: Accelerating diffusion transformer via layer caching. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 133282–133304, 2024b.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating diffusion models for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15762–15772, 2024c.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2022.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14297–14306, 2023.

Taehong Moon, Moonseok Choi, Eunggu Yun, Jongmin Yoon, Gayoung Lee, Jaewoong Cho, and Juho Lee. A simple early exiting framework for accelerated sampling in diffusion models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235, pp. 36212–36228, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, 2023.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky T. Q. Chen. Multisample flow matching: Straightening flows with minibatch couplings. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202, pp. 28100–28127, 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your steps: Optimizing sampling schedules in diffusion models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235, pp. 42947–42975, 2024.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Raphael Gontijo-Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 36479–36494, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision (ECCV)*, volume 15144 of *Lecture Notes in Computer Science*, pp. 87–103, 2024.

Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 4263–4276, 2023.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021b.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202, pp. 32211–32252, 2023.

Wenhao Sun, Ji Li, and Zhaoqiang Liu. Just-in-Time: Training-free spatial acceleration for diffusion transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 40643–40652, 2026.

Junwen Tan, Jinglin Liang, Hongyuan Chen, and Shuangping Huang. VDE: Training-free accelerating rectified flow model via velocity decomposition and estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 37918–37928, 2026.

Yixiao Wang, Ting Jiang, Zishan Shao, Hancheng Ye, Jingwei Sun, Mingyuan Ma, Qinsi Wang, Jianyi Zhang, Yiran Chen, and Hai Li. ZEUS: Accelerating diffusion models with only second-order predictor. In *Proceedings of the 34th ACM International Conference on Multimedia (ACM MM)*, 2026.

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.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. ImageReward: Learning and evaluating human preferences for text-to-image generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 15903–15935, 2023a.

Yilun Xu, Mingyang Deng, Xiang Cheng, Yonglong Tian, Ziming Liu, and Tommi Jaakkola. Restart sampling for improving generative processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 76806–76838, 2023b.

Shuchen Xue, Zhaoqiang Liu, Fei Chen, Shifeng Zhang, Tianyang Hu, Enze Xie, and Zhenguo Li. Accelerating diffusion sampling with optimized time steps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8292–8301, 2024.

Hancheng Ye, Jiakang Yuan, Renqiu Xia, Xiangchao Yan, Tao Chen, Junchi Yan, Botian Shi, and Bo Zhang. Training-free adaptive diffusion with bounded difference approximation strategy. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 306–332, 2024.

Xuanhua Yin, Yuxuan Jia, Chuanzhi Xu, and Weidong Cai. SAFE-DiT: Semantics-aware fast-path execution for high-resolution diffusion transformers, 2026a. arXiv preprint arXiv:2606.29360.

Xuanhua Yin, Chuanzhi Xu, Shunqi Mao, Wei Guo, and Weidong Cai. DefaultShift: Auditing semantic default shift in accelerated text-to-image models, 2026b. arXiv preprint arXiv:2608.21784.

Xuanhua Yin, Chuanzhi Xu, Haoxian Zhou, Boyu Wei, and Weidong Cai. AccelAes: Accelerating diffusion transformers for training-free aesthetic-enhanced image generation. In *European Conference on Computer Vision (ECCV)*, 2026c.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation. *Transactions on Machine Learning Research*, 2022.

Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. MagicBrush: A manually annotated dataset for instruction-guided image editing. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 31428–31449, 2023.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *International Conference on Learning Representations (ICLR)*, 2023.

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 (CVPR)*, pp. 586–595, 2018.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. UniPC: A unified predictor-corrector framework for fast sampling of diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 49842–49869, 2023.

Wenliang Zhao, Minglei Shi, Xumin Yu, Jie Zhou, and Jiwen Lu. FlowTurbo: Towards real-time flow-based image generation with velocity refiner. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 4148–4176, 2024.

Le Zhuo, Ruoyi Du, Han Xiao, Yangguang Li, Dongyang Liu, Rongjie Huang, Wenze Liu, Xiangyang Zhu, Fu-Yun Wang, Zhanyu Ma, Xu Luo, Zehan Wang, Kaipeng Zhang, Lirui Zhao, Si Liu, Xiangyu Yue, Wanli Ouyang, Yu Qiao, Hongsheng Li, and Peng Gao. Lumina-Next: Making Lumina-T2X stronger and faster with Next-DiT. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 131278–131315, 2024.

A Controller Specification and Error Analysis

This appendix specifies the controller, proves the identity between the proposal and correction, and separates the error of an accepted segment into carry error and field variation over the remaining span.

A.1 Exact Always-Carry Algorithm

Algorithm 1 specifies the implemented guided-field controller. The symbol $\perp$ denotes an empty carry, b is the carry buffer, N is the NFE counter, and k_* is the selected advance length. Each invocation of f counts as one NFE. A batched call that jointly evaluates the conditional and unconditional branches counts as one guided NFE, so the count tracks sequential guided evaluations rather than examples within the CFG batch. Equation (5) defines the clipped, deduplicated candidate set.

For $K_a = \min(K, T - i) \geq 3$, the descending, deduplicated candidate set is:

$$\mathcal{C}_i = \mathrm{sort}_{\downarrow}\left\{ K_a, \max\left(3, \left\lfloor \frac{3K_a}{4} \right\rfloor\right), \max\left(3, \left\lfloor \frac{K_a}{2} \right\rfloor\right), \max\left(3, \left\lfloor \frac{K_a}{4} \right\rfloor\right)\right\}, \tag{5}$$

where K_a is the clipped number of remaining schedule intervals and $\mathrm{sort}_{\downarrow}$ orders the distinct candidates from longest to shortest. Both stabilizers in Eq. (4) are 10^{-8}, and RMS residuals are computed in float32.

For the conditional-branch diagnostic, each batched CFG call stores the conditional, unconditional, and guided velocities. The selected branch supplies both residuals, while the guided velocity supplies the state update. The default controller instead scores the guided field and otherwise follows Algorithm 1.

Evaluation Accounting. Let M be the number of regular iterations with $K_a \geq 3$, and let F count all loop iterations, regular or tail, that start without a carry. Equation (6) gives the exact count:

$$N_{\mathrm{ReAL}} = M + F, \tag{6}$$

where N_{ReAL} is the total number of guided backbone evaluations. For $T \geq 3$, initialization contributes one to F. If the regular loop leaves $r \in \{1, 2\}$ tail intervals, the first tail interval consumes the existing carry and the remaining $r - 1$ intervals require fresh evaluations. Thus $F = 1 + \max(0, r - 1)$, with $r = 0$ when the last segment reaches T. If $T < 3$, there is no regular iteration and every step requires a fresh evaluation. These counts include failed verification attempts. Wall-clock runtime also includes encoding, decoding, tensor operations, and execution overhead.

A.2 Proposal and Correction Identity with Span Scaling

For arbitrary u_i, write $R_{i,k} = \sigma_{i+k} - \sigma_{i+1}$. The raw endpoint is $x_i + (\Delta\sigma_i + R_{i,k})u_i$, and the corrected endpoint is $x_i + \Delta\sigma_i u_i + R_{i,k} w_i$. Subtraction immediately gives Eq. (3). Taking the RMS norm gives Eq. (7):

$$\|\hat{x}_{i+k|i} - \tilde{x}_{i+k|i}\|_{\mathrm{rms}} = s_{i,k}\|u_i - w_i\|_{\mathrm{rms}}, \tag{7}$$

where $s_{i,k} = |R_{i,k}|$ is the residual schedule span and $\|\cdot\|_{\mathrm{rms}}$ is the state-wise RMS norm. The equality holds for any stored velocity and requires no smoothness assumption. Dividing both sides by $(h_i + \epsilon_\sigma)(\|u_i\|_{\mathrm{rms}} + \epsilon_v)$ gives $r^{\mathrm{seg}}_{i,k}$. A denominator proportional to $s_{i,k}$ would cancel the candidate-length dependence, so the implemented score instead uses the first schedule interval as its reference scale.

Disagreement between the proposal and correction captures one part of the approximation error. Both proposals inherit stored-velocity error, and velocities may change beyond the lookahead. Equation (8) shows how the local score combines direction and magnitude. For nonzero vectors, let $a = \|w_i\|_2 / \|u_i\|_2$ and let θ be their Euclidean angle. Then:

$$\frac{\|u_i - w_i\|_2^2}{\|u_i\|_2^2} = 1 + a^2 - 2a\cos\theta. \tag{8}$$

Thus, the relative residual before adding ϵ_v depends on both velocity direction and magnitude. The implemented RMS score has the same dependence, with ϵ_v for stability.

Algorithm 1 ReAL with Always-Carry and Exact Tail Handling

Require: Initial state x_0, schedule $\sigma_{0:T}$, $K \geq 3$, and $q_{\mathrm{loc}}, q_{\mathrm{seg}}$

$i \leftarrow 0$
$b \leftarrow \perp$
$N \leftarrow 0$
while $i < T$ **do**
  if $b = \perp$ **then**
    $u_i \leftarrow f(x_i, \sigma_i)$
    $N \leftarrow N + 1$
  else
    $u_i \leftarrow b$
    $b \leftarrow \perp$ ▷ Consume the stored velocity
  end if
  $K_a \leftarrow \min(K, T - i)$
  $z_i \leftarrow x_i + \Delta\sigma_i u_i$
  if $K_a < 3$ **then**
    $x_{i+1} \leftarrow z_i$
    $i \leftarrow i + 1$ ▷ No carry is produced in the tail
  else
    $w_i \leftarrow f(z_i, \sigma_{i+1})$
    $N \leftarrow N + 1$
    Compute r_i^{loc} using Eq. (4)
    $k_* \leftarrow 1$
    if $r_i^{\mathrm{loc}} \leq q_{\mathrm{loc}}$ **then**
      for k in $\mathcal{C}_i$ from Eq. (5), in descending order **do**
        if $r_{i,k}^{\mathrm{seg}} \leq q_{\mathrm{seg}}$ **then**
          $k_* \leftarrow k$
          break
        end if
      end for
    end if
    if $k_* = 1$ **then**
      $x_{i+1} \leftarrow z_i$
    else
      $x_{i+k_*} \leftarrow z_i + (\sigma_{i+k_*} - \sigma_{i+1}) w_i$
    end if
    $b \leftarrow w_i$
    $i \leftarrow i + k_*$ ▷ Carry after acceptance and fallback
  end if
end while
return x_T, N

A.3 Separating Carry Error from Segment Freezing Error

Consider a dense reference restarted from the same current state, with $y_i = x_i$ and $y_{j+1} = y_j + \Delta\sigma_j f(y_j, \sigma_j)$. Define the current carry error as $\delta_i = u_i - f(x_i, \sigma_i)$. Equation (9) gives the first-step difference:

$$z_i - y_{i+1} = \Delta\sigma_i \delta_i. \tag{9}$$

This gives the condition for an exact first dense step. A fresh evaluation gives $\delta_i = 0$. After fallback, the next carry is aligned because $w_i = f(z_i, \sigma_{i+1})$. After a longer accepted segment, the carry extrapolates the lookahead velocity to the accepted endpoint.

Fallback aligns the next velocity with the accepted state while retaining state error accumulated in reaching z_i. Refreshing after every accepted segment removes the next carry mismatch but costs another evaluation. A separate carry-policy experiment measures this trade-off.

Let $g_{i+1} = f(y_{i+1}, \sigma_{i+1})$. Summing the restarted dense updates and subtracting them from the corrected endpoint yields the exact decomposition:

$$\tilde{x}_{i+k|i} - y_{i+k} = \Delta\sigma_i \delta_i + R_{i,k}(w_i - g_{i+1}) + D_{i,k}, \tag{10}$$

$$D_{i,k} = \sum_{j=i+1}^{i+k-1} \Delta\sigma_j \big[g_{i+1} - f(y_j, \sigma_j)\big], \tag{11}$$

where $R_{i,k} = \sigma_{i+k} - \sigma_{i+1}$, g_{i+1} is the dense velocity at the restarted lookahead state, and $D_{i,k}$ is the residual-span freezing error. The three terms are the immediate carry error, its effect on the lookahead evaluation, and the residual-span error from freezing the fresh velocity along the restarted dense trajectory. The decomposition holds for any u_i and carry-error magnitude.

To express the carry contribution, suppose $f(\cdot, \sigma_{i+1})$ is continuously differentiable on the line between y_{i+1} and z_i. The mean Jacobian on that line, $\bar{J}_{i+1}$, satisfies $w_i - g_{i+1} = \bar{J}_{i+1}\Delta\sigma_i\delta_i$. Equation (10) then gives Eq. (12):

$$\tilde{x}_{i+k|i} - y_{i+k} = \Delta\sigma_i(I + R_{i,k}\bar{J}_{i+1})\delta_i + D_{i,k}, \tag{12}$$

where I is the identity operator and $\bar{J}_{i+1}$ is the mean Jacobian of the guided field along the lookahead displacement. If $\|\bar{J}_{i+1}\|_{2\to2} \le L$, the carry contribution is at most $h_i(1 + s_{i,k}L)\|\delta_i\|_2$, where $\|\bar{J}_{i+1}\|_{2\to2}$ is the Euclidean operator norm and L is its bound. The term $D_{i,k}$ is the residual-span error from Eq. (11). The controller uses the observed residual in Eq. (4). This one-segment decomposition starts from the current state, which already contains earlier errors. Its terms describe direct carry error, local amplification, and field variation over the residual span.

What the Next Residual Observes. The numerator of the next verification test is also affected by carry. At the current index, Eq. (13) gives the exact identity:

$$u_i - w_i = \delta_i + \big[f(x_i, \sigma_i) - g_{i+1}\big] - \big[w_i - g_{i+1}\big]. \tag{13}$$

This identity separates stored-velocity error, fresh dense-step velocity variation, and displacement-induced lookahead error. These vector terms can reinforce or cancel. The residual therefore combines carry mismatch with local field variation, which motivates evaluating output fidelity alongside controller diagnostics.

B Video-Generation and Editing Protocols

Hardware and Execution. All video and editing runs use the single NVIDIA H100 stated in Section 4.1, with an external batch size of one. HunyuanVideo uses 16-bit floating-point (fp16) arithmetic. Step1X-Edit stores weights in 8-bit floating point (FP8) and uses bfloat16 autocast. Each method and operating point has three untimed warm-up generations. Tables 2 and 5 report median event latency. Each speedup uses the dense run from the same task, hardware, precision, and timing session.

Metrics and Compute. Video evaluation uses VBench v0.1.5 and its 16-dimension aggregate. LPIPS uses AlexNet v0.1. Step1X-Edit quality is scored by the GPT-4o-2024-11-20 judge at temperature zero. Each editing dimension is judged three times, and its median is retained. The

five seeds are $\{11, 23, 47, 89, 157\}$. Each configuration covers 4,730 videos or 3,030 edited images. PSNR, SSIM, and LPIPS compare paired outputs with dense references. Video FLOPs and FLOPs ratios use the same-task dense reference. FLOPs and latency complement NFE because partial-network caches can have different per-evaluation costs. Appendix B.2 gives the 16 dimensions.

Controller and Operating Points. ReAL uses the shared-lookahead controller in Algorithm 1, with velocity carry after acceptance and fallback. Table 4 specifies the operating points. The video export records efficiency, quality, run identifiers, and configurations for SeaCache, VDE, ZEUS, AdaptiveDiffusion, and JiT.

Table 4: Operating points. Video columns denote medium and high acceleration. Editing columns denote Tiers 1, 2, and 3. Baseline settings use method-specific text labels from the experiment records, and these labels have no shared mathematical meaning across rows. N/A indicates an unreported point.

Method	Video medium	Video high	Editing Tier 1	Editing Tier 2	Editing Tier 3
TeaCache	ell=.31	ell=.40	ell=.20	ell=.32	ell=.41
TaylorSeer	N=4	N=5	N=3	N=4	N=5
DiCache	t=.71	t=.80	t=.60	N/A	N/A
SpeCa	ts=3	ts=6	ts=2	ts=4	ts=6
DPCache	K=13	K=9	K=16	K=11	K=8
Spectrum	ws=2	ws=4	ws=2	ws=3	ws=5
FlowCast	tau_mse=.14, s_max=3	tau_mse=.22, s_max=4	tau_mse=.10, s_max=2	N/A	N/A
SeaCache	delta=.24	delta=.33	N/A	N/A	N/A
VDE	tau_v=.18	tau_v=.27	N/A	N/A	N/A
ZEUS	alpha=.55, k=3	alpha=.70, k=5	N/A	N/A	N/A
AdaptiveDiffusion	skip_thr=.09	skip_thr=.15	N/A	N/A	N/A
JiT	r=.40, w=2	r=.55, w=3	r=.45	N/A	N/A
ReAL $(K, q_{\text{loc}}, q_{\text{seg}})$	$(12, .82, 1.60)$	$(18, .80, 1.60)$	$(8, .05, .10)$	$(12, .05, .10)$	$(12, .08, .16)$

Result Provenance. The source package records the results, all 16 VBench dimensions, and provenance hashes. The video comparison has 28 configurations: two shared references plus two tiers for ReAL and twelve accelerators. The HunyuanVideo source report gives run identifiers, all Table 2 fields, configurations, and summary provenance. Raw per-sample metrics, timings, FLOPs, and controller traces are stored on the experiment server.

B.1 Complete Editing Results

Table 5 reports all three quality scores, latency, speedup, NFE, and peak memory for Figure 5. Each acceleration tier fixes its configuration across examples and seeds. The available result summary contains point estimates rather than seed-level intervals.

B.2 VBench Dimension Scores

Tables 6 through 9 report all 16 dimensions at both operating points. Dimension scores use the reported values multiplied by 100. The reported total uses per-dimension normalization from the minimum to the maximum, a half weight for dynamic degree within the seven quality dimensions, equal weights within the nine semantic dimensions, and a 4:1 combination of quality and semantic scores. The dimension tables and the aggregate therefore use different scales. Figure 7 retains a selected ten-dimension radar layout. These four tables cover all 16 supplied dimensions. A high consistency score alone does not imply better motion. At the high-acceleration point, ReAL retains a dynamic-degree score of 66.94 versus 71.11 for dense sampling and 43.33 for direct step reduction.

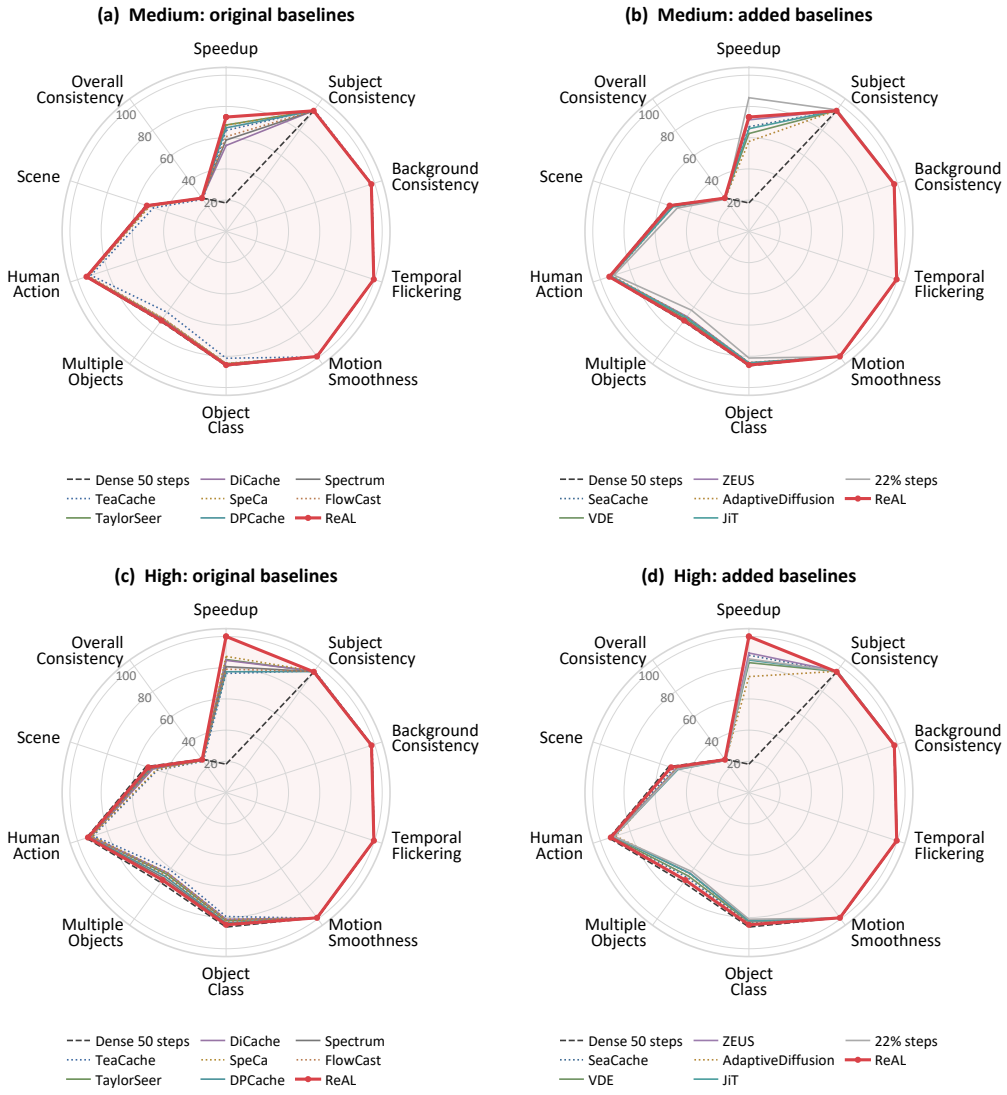

Figure 7: **VBench dimension profiles at medium and high acceleration.** Columns separate the original seven competing accelerators from the five additions. The same dense and ReAL references appear in both. Quality axes show raw dimension scores multiplied by 100 without per-axis rescaling. The speed axis uses $100\% = 5.49\times$. All panels share axis order and limits. The complete 16-dimensional tables also report dynamic degree and imaging quality. Curves show the reported dimension scores. The export provides configuration-level aggregates rather than per-seed intervals.

Table 5: **Step1X-Edit on GEdit-Bench.** Evaluation of the current ReAL controller: 606 examples, five seeds, 512^2. G_{SC}, G_{PQ}, and G_{O} denote semantic consistency, perceptual quality, and overall score. Superscripts 1, 2, and 3 denote acceleration tiers. Bold marks the best accelerated overall score within each numbered tier. FlowCast's † marks its non-accelerating setting. Latency is median event time, and Mem is peak allocated memory.

Method	Efficiency				Editing quality		
	Latency (s)↓	Speedup↑	NFE↓	Mem (GB)↓	G_{SC} ↑	G_{PQ} ↑	G_{O} ↑
Dense	6.842	1.00×	28.00	12.04	5.24	5.31	4.20
TaylorSeer^1	2.321	2.95×	9.21	12.24	4.93	4.81	4.05
TeaCache^1	2.293	2.98×	9.40	12.75	4.89	4.97	3.96
SpeCa^1	2.720	2.52×	10.90	12.30	4.99	4.94	4.00
Spectrum^1	1.734	3.95×	6.80	12.26	4.83	5.06	4.00
DPCache^1	2.354	2.91×	9.38	12.27	5.13	4.95	4.19
DiCache^1	2.887	2.37×	11.39	12.27	4.70	4.81	3.92
JiT^1	1.690	4.05×	6.71	12.67	4.88	4.79	3.96
ReAL^1	1.773	3.86×	7.40	12.46	5.10	5.23	**4.24**
TaylorSeer^2	1.829	3.74×	7.28	12.60	4.94	5.02	4.07
TeaCache^2	1.865	3.67×	7.65	12.11	5.03	4.93	4.03
SpeCa^2	1.889	3.62×	7.84	12.46	4.96	4.81	4.02
Spectrum^2	1.460	4.69×	5.77	12.26	4.86	4.82	3.99
DPCache^2	1.671	4.09×	6.91	12.67	4.90	4.88	4.00
ReAL^2	1.494	4.58×	6.20	12.54	5.11	5.19	**4.21**
TaylorSeer^3	1.432	4.78×	5.76	12.65	5.04	5.05	**4.16**
TeaCache^3	1.419	4.82×	5.77	12.63	4.93	4.98	4.07
SpeCa^3	1.571	4.36×	6.50	12.42	5.03	5.08	4.13
Spectrum^3	1.359	5.04×	5.68	12.30	4.87	5.04	4.04
DPCache^3	1.374	4.98×	5.81	12.37	5.02	5.12	4.05
ReAL^3	1.311	5.22×	5.50	12.41	5.06	5.14	4.11
FlowCast†	7.699	0.89×	30.98	12.18	4.64	4.72	3.98

Table 6: **Medium acceleration: VBench dimensions (a).** Raw dimension scores ×100. Dense and direct step reduction are shared references.

Method	Subj.	Bg.	Flicker	Smooth	Dynamic	Aesthetic	Imaging	Object
Dense 50 steps	95.36	97.76	99.44	98.99	71.11	60.36	70.10	86.10
22% steps	96.15	98.26	99.56	99.31	43.33	58.50	67.05	81.11
TeaCache	96.12	98.23	99.67	99.35	57.50	58.59	67.22	81.27
TaylorSeer	95.58	97.91	99.52	99.11	66.67	59.80	69.27	84.84
DiCache	95.58	97.87	99.51	99.08	67.22	60.02	69.73	84.98
SpeCa	95.61	97.95	99.54	99.13	64.17	59.82	69.22	84.72
DPCache	95.48	97.86	99.49	99.06	68.61	60.11	69.59	85.40
Spectrum	95.42	97.81	99.47	99.03	70.00	60.23	69.88	85.77
FlowCast	95.66	97.97	99.55	99.19	65.00	59.80	69.13	84.50
SeaCache	95.53	97.88	99.51	99.08	68.06	60.01	69.35	85.23
VDE	95.60	97.93	99.56	99.16	65.28	59.84	69.29	84.62
ZEUS	95.70	97.99	99.56	99.15	64.44	59.71	69.20	84.15
AdaptiveDiffusion	95.74	97.99	99.57	99.18	61.67	59.42	68.90	83.99
JiT	95.82	97.99	99.56	99.15	63.06	59.69	69.15	84.23
ReAL	95.46	97.82	99.48	99.04	69.44	60.19	69.83	85.62

Table 7: **Medium acceleration: VBench dimensions (b).** Raw dimension scores $\times 100$. Dense and direct step reduction are shared references.

Method	Multiple	Action	Color	Spatial	Scene	App. style	Temp. style	Overall cons.
Dense 50 steps	71.19	94.20	88.70	68.68	53.88	19.80	23.89	26.44
22% steps	62.44	90.80	84.51	61.49	48.27	18.99	22.96	25.41
TeaCache	64.06	91.00	84.74	61.97	48.88	19.13	22.97	25.31
TaylorSeer	69.39	93.40	87.68	66.98	52.55	19.62	23.63	26.15
DiCache	69.52	93.60	87.87	67.51	52.76	19.66	23.70	26.25
SpeCa	68.86	93.20	87.31	66.69	51.56	19.58	23.60	26.08
DPCache	70.04	93.60	88.00	67.69	53.05	19.70	23.75	26.28
Spectrum	70.68	94.20	88.41	68.25	53.49	19.74	23.80	26.36
FlowCast	68.71	92.60	87.18	66.54	51.90	19.53	23.55	26.08
SeaCache	69.77	93.80	87.89	67.47	52.70	19.65	23.70	26.21
VDE	69.11	93.40	87.32	66.79	51.88	19.58	23.59	26.10
ZEUS	67.43	92.80	86.72	66.21	51.55	19.51	23.51	26.02
AdaptiveDiffusion	67.70	92.20	86.59	65.85	51.19	19.48	23.48	25.94
JiT	68.19	92.80	86.87	65.57	51.01	19.50	23.51	26.02
ReAL	70.57	94.00	88.23	68.14	53.35	19.72	23.79	26.33

Table 8: **High acceleration: VBench dimensions (a).** Raw dimension scores $\times 100$. Dense and direct step reduction are shared references.

Method	Subj.	Bg.	Flicker	Smooth	Dynamic	Aesthetic	Imaging	Object
Dense 50 steps	95.36	97.76	99.44	98.99	71.11	60.36	70.10	86.10
22% steps	96.15	98.26	99.56	99.31	43.33	58.50	67.05	81.11
TeaCache	96.38	98.38	99.68	99.41	50.00	57.92	66.25	79.36
TaylorSeer	96.02	98.16	99.63	99.26	57.22	58.47	67.56	82.00
DiCache	96.19	98.20	99.62	99.27	55.83	58.84	67.54	81.15
SpeCa	96.13	98.21	99.63	99.30	48.33	58.64	67.07	81.16
DPCache	95.82	98.10	99.60	99.20	61.11	59.34	68.39	83.38
Spectrum	95.76	98.05	99.58	99.19	63.61	59.46	68.83	83.94
FlowCast	95.97	98.13	99.61	99.31	58.33	59.07	67.91	82.60
SeaCache	95.82	98.05	99.59	99.20	61.11	59.37	68.36	83.66
VDE	95.90	98.14	99.63	99.30	57.50	59.13	68.11	82.80
ZEUS	96.07	98.21	99.63	99.27	56.39	58.78	67.47	82.28
AdaptiveDiffusion	96.04	98.18	99.62	99.29	50.56	58.64	67.49	82.16
JiT	96.22	98.23	99.64	99.29	54.44	58.76	67.66	82.08
ReAL	95.62	97.93	99.54	99.12	66.94	59.85	69.20	84.65

Table 9: **High acceleration: VBench dimensions (b).** Raw dimension scores $\times 100$. Dense and direct step reduction are shared references.

Method	Multiple	Action	Color	Spatial	Scene	App. style	Temp. style	Overall cons.
Dense 50 steps	71.19	94.20	88.70	68.68	53.88	19.80	23.89	26.44
22% steps	62.44	90.80	84.51	61.49	48.27	18.99	22.96	25.41
TeaCache	61.07	89.40	82.03	60.26	46.35	18.72	22.56	24.83
TaylorSeer	64.60	91.00	85.20	63.26	49.61	19.19	23.09	25.47
DiCache	63.56	91.20	85.24	63.26	49.25	19.20	23.06	25.52
SpeCa	63.61	90.40	84.17	62.22	47.07	19.04	22.89	25.36
DPCache	66.29	92.20	85.62	64.13	50.71	19.40	23.26	25.83
Spectrum	67.64	92.60	86.48	65.25	50.95	19.39	23.28	25.89
FlowCast	65.60	91.20	85.49	64.06	50.13	19.30	23.15	25.67
SeaCache	67.18	92.00	86.33	65.14	50.60	19.41	23.35	25.80
VDE	65.99	92.00	85.81	64.23	49.03	19.28	23.17	25.71
ZEUS	63.95	91.00	84.30	63.29	49.06	19.23	23.10	25.49
AdaptiveDiffusion	64.33	89.80	84.56	62.73	49.07	19.19	23.07	25.51
JiT	64.05	91.00	84.39	61.22	47.82	19.17	23.04	25.51
ReAL	69.26	93.00	87.33	67.08	52.39	19.59	23.56	26.08

C Additional Experimental Detail

C.1 Evaluation Protocols and Metrics

All analysis figures use frozen experimental summaries whose source hashes appear in the accompanying manifest. Standard DrawBench sweeps use the first 100 prompts and seed $42 + i$, where i is the prompt index. The expanded SD3 experiment uses 200 prompts with the same rule. The PartiPrompts experiment uses 150 prompts, with 98 outside the Basic and Simple Detail categories and 52 within them. Prompt groups follow the supplied challenge annotations.

The SD3-family runs use bfloat16 inference and float32 residuals. The newer FLUX timing experiment uses diffusers fp16. ImageReward measures prompt-conditioned preference and is averaged across paired outputs. Retention divides method mean ImageReward by dense mean ImageReward. Since ImageReward is signed, this ratio can fall below zero or exceed 100%. LPIPS measures agreement with the matched dense image, CLIPScore measures alignment between text and image, and PickScore estimates human image preference (Kirstain et al., 2023).

For CFG, one forward call evaluates both conditioning branches in a batch, and NFE counts that batched backbone call once. Its FLOPs depend on the architecture and guidance configuration. For a fixed backbone with full-layer execution, lower NFE means fewer full network calls. Latency also depends on memory, precision, device load, and implementation, so each runtime comparison uses its own dense reference and timing session.

Baseline Implementations. The dense-head rule uses the controller's carry and corrected update with a fixed horizon. A setting $m{:}k$ takes m exact dense steps, then uses a fixed span of k intervals. Fixed stride instead reevaluates at each stride boundary and freezes that velocity. SD3 TeaCache uses recent output changes to decide reuse, and SD3 TaylorSeer predicts outputs with a second-order rule. Both operate at the model-output level. AdaptiveDiffusion uses a third-order latent-difference test rescaled for flow updates, with at most four consecutive skipped predictions by default and eight in additional FLUX runs. In the tables, ReAL (guided) computes both verification residuals from the guided CFG field. ReAL (cond.) computes both residuals from the conditional CFG branch but still updates the state with the guided field.

Trace and Replay Accounting. Decision traces cover regular segment starts. Figure 10c averages their selected lengths within each bin and treats tail steps separately. The geometric ablation replays recorded segmentations under its stated carry or refresh policy. Figure 10b labels ReAL with the controller cost of 14.76 NFE. The diagnostic replay averages 15.18 NFE because it evaluates an unused terminal velocity on 42 of 100 trajectories. Its outputs provide the drift and local-discrepancy values.

Controller Interpretation and Transfer. The residual measures the endpoint difference implied by u_i and w_i over a proposed span. Later field changes and carry error determine how this difference relates to final quality. Thresholds and the span cap are therefore selected on the target model and schedule using task quality and end-to-end latency. Fallback rate, selected spans, and accepted residuals describe controller behavior. Few-step schedules and encoding or decoding bottlenecks leave less room for speedup. The three tasks use their own models, schedules, metrics, and operating points, so the experiments evaluate transfer of the controller structure with workload-specific settings. Whether acceleration preserves semantic default distributions across repeated samples remains a separate evaluation question (Yin et al., 2026b).

C.2 Statistical Comparisons

Paired comparisons resample prompt indices while preserving each prompt and seed pair across methods. We draw 20,000 bootstrap samples with random seed 20260913 and report percentile 95% confidence intervals for the mean difference. Standard errors equal the sample standard deviation of paired differences divided by $\sqrt{n}$, where n is the number of paired observations. The intervals describe variation over prompts under the fixed seed assignment and are unadjusted for exploratory comparisons. A confidence interval that includes zero does not establish the sign of the population

mean difference. Tables report NFE beside quality differences to show each comparison's compute cost.

C.3 Measured Operating Points and Paired Differences

The tables report measured configurations and paired differences. CLIP denotes CLIPScore, and a dash marks an unreported metric. The source data cover every plotted configuration, while the tables show selected operating points.

Additional Threshold and Schedule Comparisons. At $q_{\text{seg}} = 3.8$ and $K = 24$, ReAL obtains 0.738 IR at 6.41 NFE, compared with 0.034 at 5 NFE for rule 2:24. Rule 2:12 is stronger at $K = 12$ (Table 10). In the 200-prompt SD3-Medium study with $T = 15$, another ReAL point gives 0.3116 IR at 6.73 NFE, compared with 0.3019 at 7 NFE for rule 4:6. Dense IR is 0.7430. Figure 6 reports absolute mean ImageReward for the additional backbones. The source data also include output-based TaylorSeer and conditional-branch variants.

Table 10: **Segment-cap study on FLUX.** Absolute mean IR on 100 matched prompt and seed pairs. Each threshold is fixed across K. The rule and ReAL columns retain their measured NFE. The corresponding cap-sweep plot appears in Appendix C.

	$q_{\text{seg}} = 2.6$ vs. rule 3:K				$q_{\text{seg}} = 3.8$ vs. rule 2:K			
K	ReAL		Rule		ReAL		Rule	
	NFE	IR	NFE	IR	NFE	IR	NFE	IR
12	7.65	0.884	8.00	0.893	6.32	0.676	7.00	0.831
16	7.76	0.853	7.00	0.700	6.16	0.635	6.00	0.578
24	7.83	0.904	6.00	0.283	6.41	0.738	5.00	0.034

Table 11: FLUX cap comparisons. Differences are ReAL minus rule. Positive ΔNFE means more compute. Intervals are exploratory and unadjusted for operating-point selection.

K	q_{seg}	Rule	ΔNFE	ΔIR	Paired SE	95% CI
12	2.6	3:12	-0.35	-0.009	0.046	[-0.102, +0.082]
12	3.8	2:12	-0.68	-0.155	0.073	[-0.305, -0.022]
16	2.6	3:16	+0.76	+0.152	0.058	[+0.040, +0.268]
16	3.8	2:16	+0.16	+0.056	0.075	[-0.097, +0.195]
24	2.6	3:24	+1.83	+0.621	0.073	[+0.481, +0.765]
24	3.8	2:24	+1.41	+0.704	0.088	[+0.531, +0.876]

Table 12: Completed SD3-Medium evaluation with 200 DrawBench prompts. Each step count has its own dense reference.

T	Method	Setting	NFE	IR	IR ratio (%)
15	Dense	N/A	15.000	0.743	100.0
15	Rule	3:6	6.000	0.047	6.3
15	Rule	4:6	7.000	0.302	40.6
15	ReAL (guided)	$q_{\text{seg}} = 3.2$	6.245	0.198	26.6
15	ReAL (guided)	$q_{\text{seg}} = 2.2$	6.730	0.312	41.9
15	ReAL (guided)	$q_{\text{seg}} = 1.65$	7.265	0.382	51.4
15	ReAL (guided)	$q_{\text{seg}} = 1.25$	8.550	0.592	79.6
15	ReAL (guided)	$q_{\text{seg}} = 0.9$	10.305	0.650	87.4
15	ReAL (guided)	$q_{\text{seg}} = 0.6$	12.795	0.718	96.6
28	Dense	N/A	28.000	0.839	100.0
28	Rule	3:6	8.000	0.459	54.7
28	Rule	6:3	14.000	0.786	93.6
28	ReAL (guided)	$q_{\text{seg}} = 3.2$	9.025	0.494	58.9
28	ReAL (guided)	$q_{\text{seg}} = 0.9$	15.115	0.807	96.1

Table 13: SD3-Medium, DrawBench-100. TeaCache uses output-delta reuse. TaylorSeer is an output-level reimplementation.

Method	Setting	NFE	IR	CLIP	LPIPS↓	PickScore
Dense	N/A	28.00	0.736	27.48	0.000	22.06
Rule	6:3	14.00	0.678	27.39	0.096	21.98
Rule	4:4	11.00	0.598	27.40	0.182	21.85
ReAL (guided)	$q_{\text{seg}} = 0.9$	14.76	0.678	27.47	0.105	21.93
ReAL (guided)	$q_{\text{seg}} = 1.25$	12.43	0.594	27.49	0.171	21.88
ReAL (guided)	$q_{\text{seg}} = 1.65$	10.84	0.580	27.61	0.214	21.86
ReAL (cond.)	$q_{\text{seg}} = 0.45$	11.01	0.582	N/A	N/A	N/A
AdaptiveDiffusion (flow)	`delta=0.5`	13.59	0.591	N/A	N/A	21.80
TeaCache (output-delta)	`tau=0.8`	13.05	0.652	N/A	N/A	21.98
TeaCache (output-delta)	`tau=1.2`	10.64	0.586	N/A	N/A	21.82
TaylorSeer (output)	`r=1`	17.00	0.653	N/A	N/A	21.98

Table 14: SD3.5-Large, DrawBench-100, with the same implementation labels as Table 13.

Method	Setting	NFE	IR	CLIP	LPIPS↓	PickScore
Dense	N/A	28.00	0.879	27.78	0.000	22.96
Rule	3:6	8.00	0.755	27.95	0.182	22.57
Rule	4:6	9.00	0.800	27.93	0.152	22.60
Rule	4:4	11.00	0.830	27.86	0.103	22.77
ReAL (guided)	$q_{\text{seg}} = 0.6$	11.87	0.820	27.89	0.098	22.81
ReAL (guided)	$q_{\text{seg}} = 1.25$	8.25	0.794	27.93	0.237	22.67
ReAL (cond.)	$q_{\text{seg}} = 0.6$	8.20	0.787	N/A	N/A	N/A
TeaCache (output-delta)	`tau=0.5`	10.96	0.844	N/A	N/A	22.79
TeaCache (output-delta)	`tau=0.8`	8.80	0.794	N/A	N/A	22.67
TeaCache (output-delta)	`tau=1.2`	7.40	0.781	N/A	N/A	22.53
TaylorSeer (output)	`r=1`	17.00	0.866	N/A	N/A	22.88

Table 15: Lumina-Next-SFT, DrawBench-100. Quality and NFE compare adaptive advancement with fixed-stride and dense-head allocation.

Method	Setting	NFE	IR	CLIP	LPIPS↓	PickScore
Dense	N/A	30.00	0.546	26.25	0.000	22.40
Rule	6:3	15.00	0.486	26.34	0.034	22.26
Rule	2:6	8.00	0.140	26.17	0.189	21.76
Rule	3:6	9.00	0.221	26.42	0.150	21.89
ReAL (guided)	$q_{\text{seg}} = 0.35$	14.29	0.405	26.41	0.070	22.16
ReAL (guided)	$q_{\text{seg}} = 0.9$	9.27	0.222	26.18	0.170	21.94
ReAL (guided)	$q_{\text{seg}} = 1.25$	7.87	0.204	26.15	0.229	21.85

Table 16: SD3-Medium on the 150-prompt PartiPrompts subset. The 28-step dense reference belongs to this prompt set.

Method	Setting	NFE	IR	CLIP	LPIPS↓	PickScore
Dense	N/A	28.00	1.065	N/A	N/A	N/A
Rule	6:3	14.00	1.032	N/A	N/A	N/A
Rule	4:4	11.00	0.976	N/A	N/A	N/A
ReAL (guided)	$q_{\text{seg}} = 1.25$	12.66	1.037	N/A	N/A	N/A
ReAL (guided)	$q_{\text{seg}} = 1.65$	11.14	1.013	N/A	N/A	N/A
ReAL (guided)	$q_{\text{seg}} = 3.2$	9.37	0.905	N/A	N/A	N/A

Table 17: FLUX NFE sweep on DrawBench-100. Quality uses this sweep's dense reference.

Method	Setting	NFE	IR	CLIP	LPIPS↓	PickScore
Dense	N/A	50.00	1.016	27.59	0.000	23.17
Rule	2:12	7.00	0.831	27.26	0.275	N/A
Rule	3:12	8.00	0.893	27.57	0.222	N/A
Rule	3:8	10.00	0.963	27.54	0.163	N/A
ReAL	$q_{\text{seg}} = 2.6$	7.65	0.884	27.46	0.300	N/A
ReAL	$q_{\text{seg}} = 3.8$	6.32	0.676	27.40	0.361	N/A
AdaptiveDiffusion (flow)	`delta=0.4`	13.90	0.922	N/A	N/A	23.11
AdaptiveDiffusion (flow)	`delta=0.8`	12.15	0.956	N/A	N/A	23.13

Table 18: Carry-policy study on FLUX, 100 prompts. Always-carry and short-only carry use $q_{\text{seg}} = 1.65$ to isolate cross-segment reuse. Fallback-only rows report additional combinations of thresholds and policies.

Policy	q_{seg}	NFE	IR	Accepted k	NFE/iteration
Always carry	1.65	9.48	0.872	6.19	1.06
Fallback-only carry	2.6	10.14	0.814	10.13	1.96
Fallback-only carry	3.6	10.00	0.793	10.66	1.89
Carry if $k \leq 4$	1.65	11.06	0.863	8.73	1.73

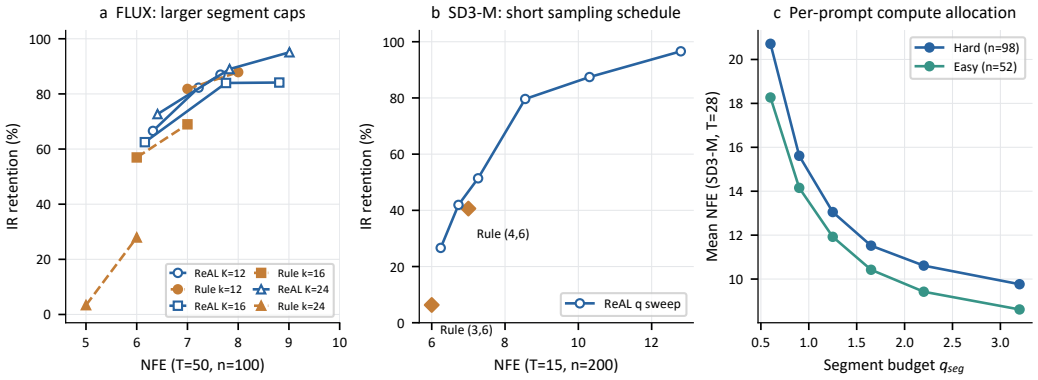

Figure 8: **Adaptive allocation across segment caps, schedules, and prompts. a**, FLUX cap sweep. Marker shape identifies K, and color identifies the method. The rule uses $m \in \{2, 3\}$ and $k = K$. **b**, Expanded 200-prompt SD3-Medium sweep at $T = 15$, with fixed-rule configurations shown at their actual NFE. **c**, On 150 PartiPrompts, the controller allocates more evaluations to the 98 prompts outside the Basic or Simple Detail categories than to the remaining 52. Each curve sweeps the segment threshold, and the prompt groups use the benchmark categories.

D THE EMPIRICAL FIXED-STRIDE ERROR RELATIONSHIP

For each fixed-stride configuration $c = (b, k)$, let $\mathcal{S}_c$ contain its evaluated segments (τ, i, j), where b is the backbone, k is the stride, τ indexes a calibration trajectory, $i \in \{0, k, 2k, \ldots\}$ is below the final index T_b, and $j = \min(i + k, T_b)$. For dense states $X_t^{b,\tau}$, dense velocities $V_t^{b,\tau}$, and schedule values σ_t^b, we define:

$$\widetilde{e}_{i,j}^{b,\tau} = \sum_{t=i}^{j-1} (\sigma_{t+1}^b - \sigma_t^b)(V_t^{b,\tau} - V_i^{b,\tau}),$$

$$E_c = \frac{1}{|\mathcal{S}_c|} \sum_{(\tau,i,j)\in\mathcal{S}_c} \frac{\|\widetilde{e}_{i,j}^{b,\tau}\|_2}{\|X_j^{b,\tau} - X_i^{b,\tau}\|_2 + 10^{-12}}, \tag{14}$$

where t indexes dense steps within a segment. Each ratio associated with one trajectory and one segment has equal weight in E_c, including a shorter final segment. Under an exact dense Euler update, $\widetilde{e}_{i,j}^{b,\tau}$ equals the fixed-stride endpoint discrepancy in Eq. (15). The reported values use the velocity-difference sum above, which matches the analysis script applied to the stored tensors. Calibration uses 100 dense FLUX trajectories and 20 for each of the three CFG backbones. The corresponding fixed-stride image metrics use 100 prompts, so calibration trajectories and configuration-level comparisons have different statistical roles.

Using E_c from Eq. (14), we fit $\log L_c = \alpha + \beta \log E_c$ to 22 configuration-level points, where L_c is the arithmetic mean of 100 matched LPIPS scores, α is the intercept, and β is the slope. The fit gives $\exp(\alpha) = 1.1613$ and $\beta = 0.5063$, with an RMS log residual of 0.0570. Four-fold leave-one-backbone-out evaluation refits both coefficients on the other three backbones. We denote the resulting held-out prediction by $\widehat{L}_c$. The accompanying comma-separated-value (CSV) file contains all selected points and held-out predictions.

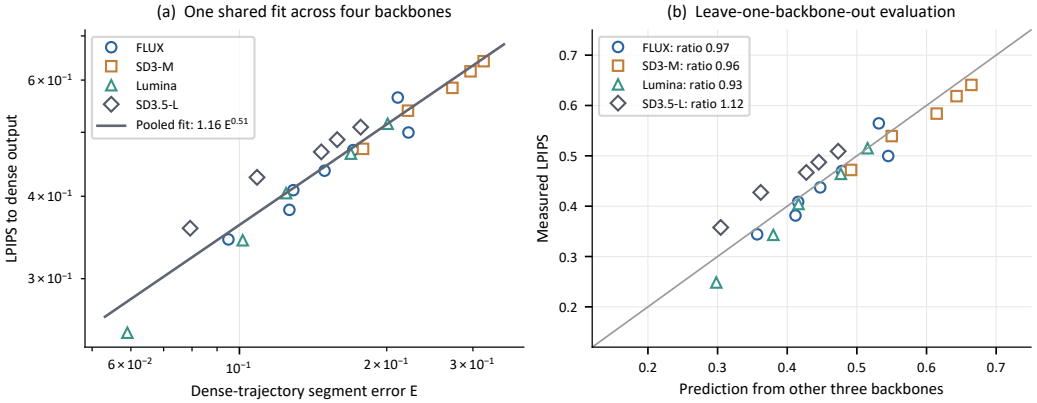

Figure 9: **An empirical fixed-stride relationship across the tested backbones. (a)**, Pooled power-law fit to 22 configuration means. **(b)**, Predictions after fitting on the other three backbones. Legend ratios summarize measured LPIPS divided by predicted LPIPS as geometric means within each held-out backbone. The largest individual relative prediction error is about 20.0% for Lumina. The fit summarizes configuration means that share prompts and dense references.

Held-out backbone	Geometric measured/predicted	Max. $\lvert\widehat{L}/L - 1\rvert$
FLUX	0.9721	9.08%
SD3-Medium	0.9634	5.25%
Lumina-Next-SFT	0.9340	20.01%
SD3.5-Large	1.1229	15.29%

The fitted relation describes LPIPS within the tested fixed-stride family. Front-loaded and adaptive segments change both the amount and placement of error. Appendix E evaluates their error propagation and allocation profiles.

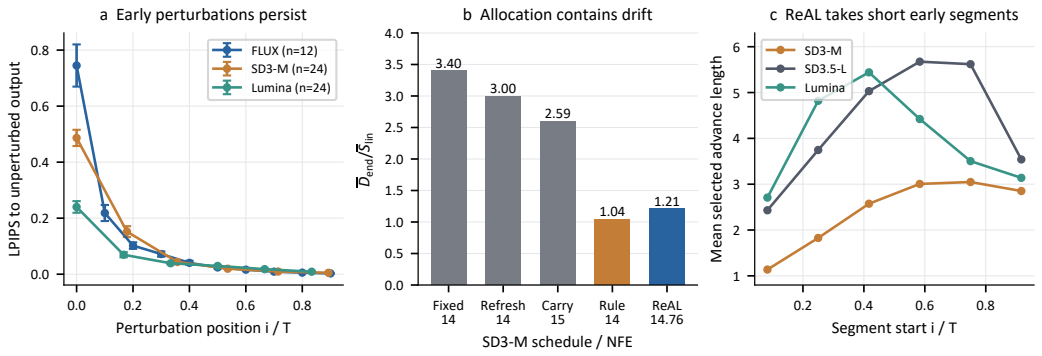

Figure 10: **ReAL adapts evaluation placement. a**, Mean LPIPS after dense continuation from a structured perturbation. Bars show one standard error across prompts. **b**, Mean terminal drift divided by mean summed dense-path discrepancy on SD3-Medium ($n = 100$). Axis labels show controller NFE. The ReAL replay adds one unused terminal query on 42 trajectories. **c**, Mean selected advance over recorded regular starts in six trajectory bins ($n = 100$). Tail steps are excluded.

E Analysis of Adaptive Segment Allocation

Why the Location of an Advance Matters. Let X_j denote the dense state and define $V_j = f(X_j, \sigma_j)$ at every dense index j. At index i, we add a perturbation with Euclidean norm $0.02\|X_i\|_2$ in the direction $V_i - V_{i+1}$, then continue with dense evaluations. Figure 10a reports LPIPS (Zhang et al., 2018) to the unperturbed output for FLUX ($n = 12$), SD3-Medium ($n = 24$), and Lumina-Next-SFT ($n = 24$), where n is the number of prompts. Early interventions cause the largest perceptual deviations, which supports shorter advances where discrepancies have greater later effects.

Compute Allocation Controls Propagated Drift. We separate local approximation error from the deviation accumulated during sampling. For a fixed stride from i to j, the dense-trajectory discrepancy is:

$$e^{\text{fix}}_{i,j} = X_j - X_i - (\sigma_j - \sigma_i)V_i, \tag{15}$$

where X_i is the dense reference state at index i and $V_i = f(X_i, \sigma_i)$ is its dense velocity. For the lookahead-corrected schedules, Eq. (16) gives the corresponding diagnostic:

$$e^{\text{pc}}_{i,j} = X_j - X_i - \Delta\sigma_i V_{c_i} - (\sigma_j - \sigma_{i+1})V_{i+1}, \tag{16}$$

where c_i indexes the carried dense velocity V_{c_i}, and V_{i+1} is the dense lookahead velocity. Refreshing at the segment start sets $c_i = i$. For a one-step advance ($j = i + 1$), the final term is omitted.

For configuration m and dense trajectory τ, let $\mathcal{G}_{m,\tau} = \{(i, j, c_i)\}$ be its visited segments and carry indices, where m identifies the schedule and c_i has the meaning given above. We use $e^{m,\tau}_{i,j} = e^{\text{fix},\tau}_{i,j}$ for a fixed-stride schedule and $e^{m,\tau}_{i,j} = e^{\text{pc},\tau}_{i,j}$ for a corrected schedule. We then define:

$$S^{m,\tau}_{\text{lin}} = \frac{\sum_{(i,j,c_i)\in\mathcal{G}_{m,\tau}} \|e^{m,\tau}_{i,j}\|_2}{\|X^\tau_T\|_2}, \qquad D^{m,\tau}_{\text{end}} = \frac{\|x^{m,\tau}_T - X^\tau_T\|_2}{\|X^\tau_T\|_2 + 10^{-12}},$$

where X^τ_T is the dense endpoint and $x^{m,\tau}_T$ is the free-running replay endpoint from the same initial latent. For ReAL, the replay fixes the recorded segment boundaries, completes the unlogged tail with one-step advances, and evaluates velocities at its own replay states. For each configuration, we take the arithmetic mean of each quantity over the 100 trajectories and report $\overline{D}^m_{\text{end}}/\overline{S}^m_{\text{lin}}$. This ratio compares free-running drift with total local discrepancy on the dense path.

On SD3-Medium, uniform stride two gives a ratio of 3.40 at 14 NFE. Corrected endpoints with a refreshed segment-start velocity or a carried lookahead velocity lower it to 3.00 and 2.59 at 14 and 15 NFE. A dense prefix gives 1.04 at 14 NFE, while ReAL gives 1.21 at a controller cost of 14.76 NFE (Figure 10b). The diagnostic replay costs 15.18 NFE because trajectories with a tail compute one unused terminal velocity. These results connect placement to drift, and ReAL selects shorter early segments (Figure 10c). The main experiments vary the cap, schedule length, and prompt. Appendix D connects dense-path segment error to perceptual deviation across four backbones.

F CURRENT IMAGE-GENERATION COMPARISONS

The FLUX.1-dev examples use a 50-step schedule, guidance 3.5, and 1024^2 resolution. Methods in each row share saved initial noise and text embeddings. Prompts were fixed before output inspection. ReAL uses $K = 12$, $q_{\text{loc}} = 0.70$, and $q_{\text{seg}} = 1.65$, with carry after acceptance and fallback. Configurations remain fixed across prompts.

TaylorSeer uses first-order prediction to fit the local memory budget, and SpeCa uses its Black Forest Labs bf16 runtime. Other main comparisons use diffusers fp16. FlowCast uses serial draft verification within the 32 GB memory budget. Images are resized only for display, and typography examples preserve the prompt text.

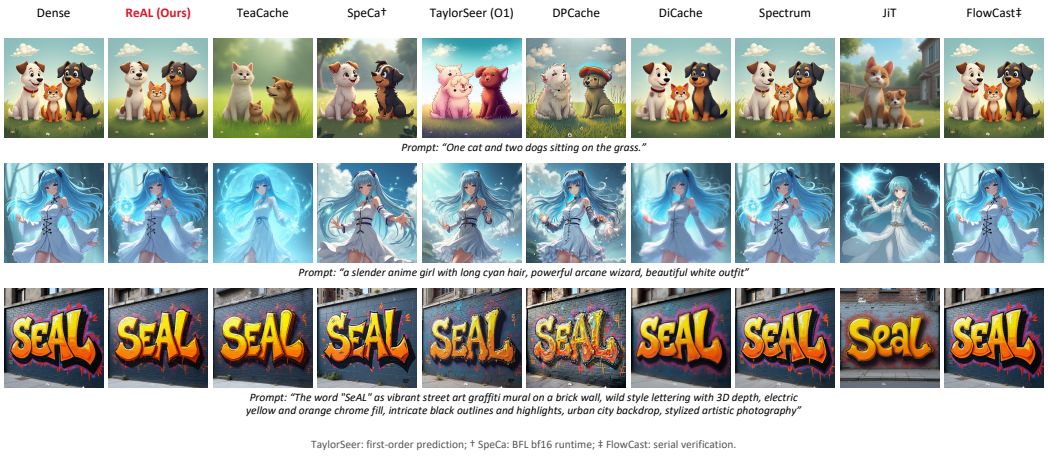

Figure 11: **Additional accelerator comparisons.** The same prompts and noise compare ReAL with the original baseline set, including SpeCa, TaylorSeer, DPCache, Spectrum, JiT, and FlowCast. Runtime environments are specified above.

G CURRENT VIDEO-GENERATION COMPARISONS

Figure 12 compares all fifteen methods and references on three HunyuanVideo prompts. Methods share initial noise (seed 11) and text conditioning for each prompt. Videos contain 65 frames at 544×960 resolution and 24 fps, with embedded guidance 6. Dense sampling uses 50 steps, and direct step reduction uses 11. ReAL uses $K = 18$, $q_{\text{loc}} = 0.80$, and $q_{\text{seg}} = 1.60$, carrying the lookahead after acceptance and fallback. These frames come from a separate matched-seed bf16 reproduction rather than the timing runs in Table 2. The source manifest records every qualitative configuration.

VDE, ZEUS, AdaptiveDiffusion, and JiT use HunyuanVideo adaptations. JiT applies the spatial selection, interpolation, and microflow functions from its public image implementation across latent frames. FlowCast uses serial draft verification. Every method keeps one configuration across the three prompts.

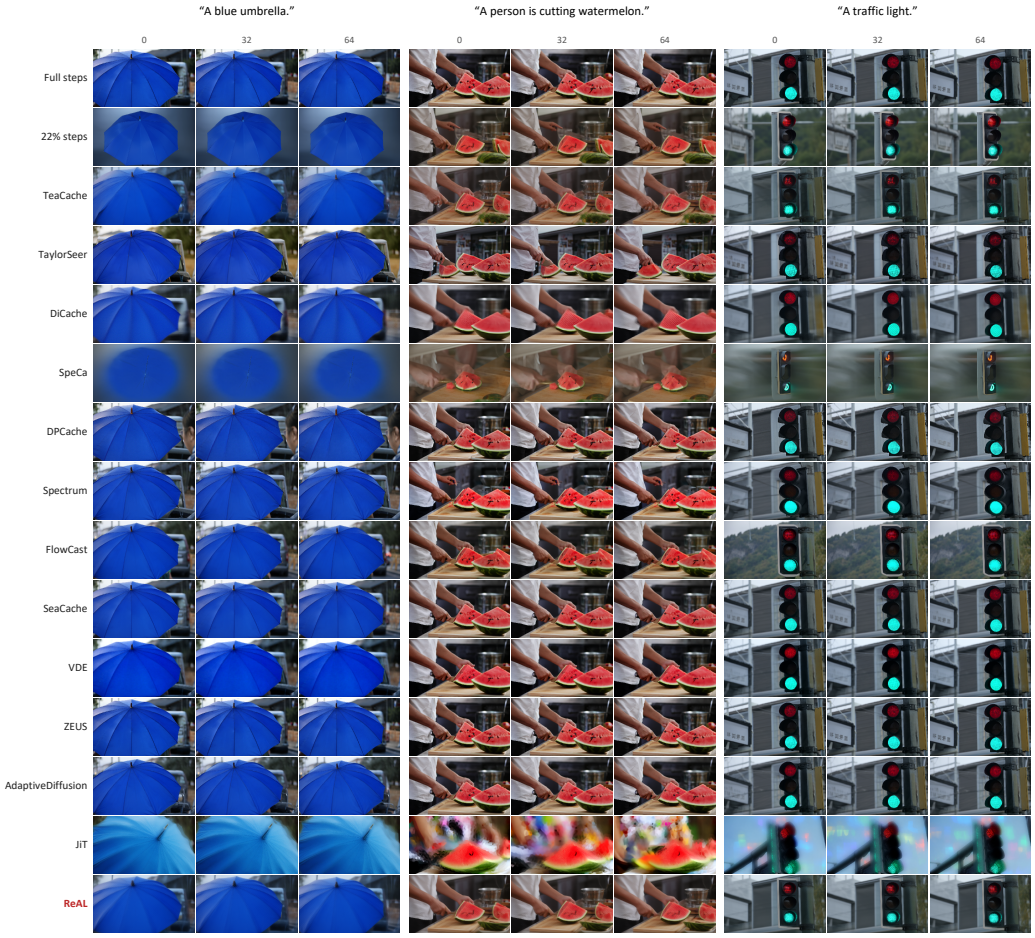

Figure 12: **Complete HunyuanVideo qualitative comparison.** Rows show dense sampling, direct step reduction, twelve acceleration baselines, and ReAL. Each prompt group displays frames 0, 32, and 64 from the same generated video. The panels retain the full frame and original colors.

H CROSS-BACKBONE IMAGE-GENERATION EXAMPLES

These examples extend the current carry-after-both-outcomes comparison from FLUX to three other backbones.

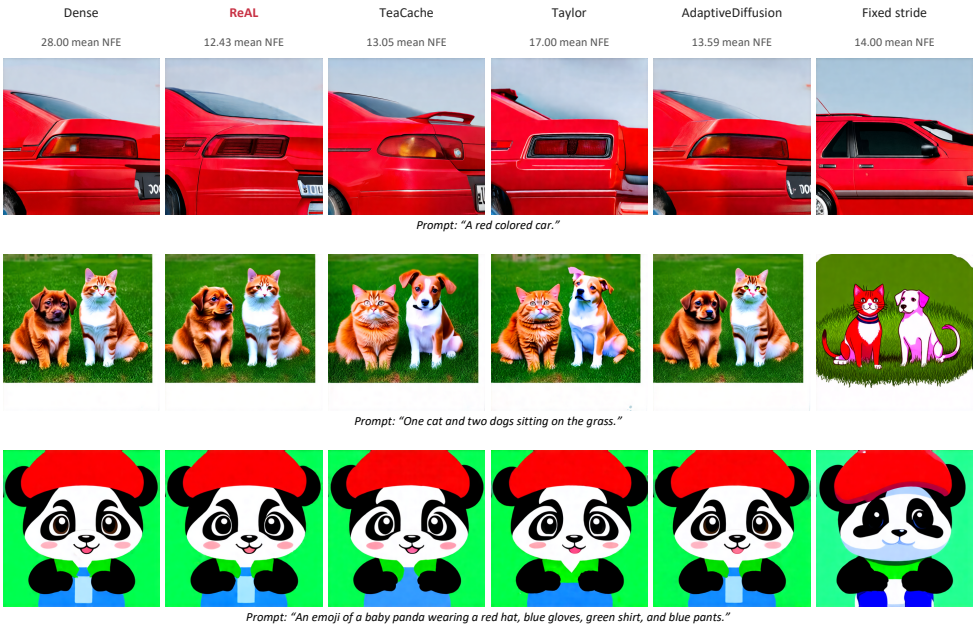

Figure 13: **SD3-Medium.** ReAL, dense sampling, output-level cache baselines, flow-adapted AdaptiveDiffusion, and fixed stride. Headers report 100-prompt mean NFE.

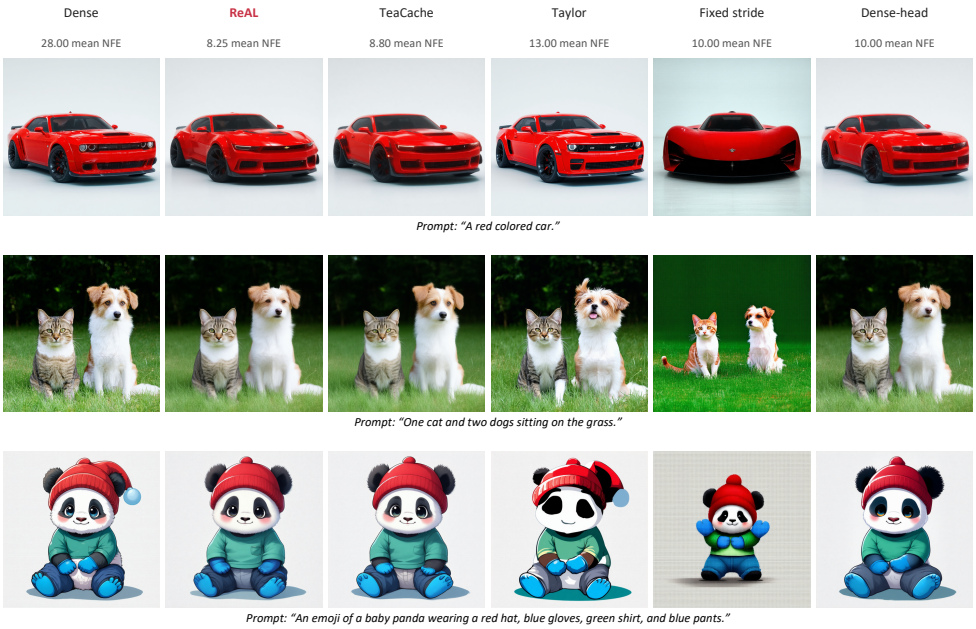

Figure 14: **SD3.5-Large.** The same prompts compare ReAL and dense sampling with output-level TeaCache and TaylorSeer, fixed stride, and a dense-head schedule.

The three DrawBench prompts test a simple object, object composition, and multiple color attributes. They use indices 0, 46, and 70 with seed 42 plus the prompt index. Every method keeps one configuration across the three prompts. Column headers give its mean NFE over the corresponding 100-prompt evaluation. Figure 6 shows the full quantitative trade-offs.

The SD3-family TeaCache and TaylorSeer comparisons operate on model outputs. TeaCache reuses outputs, TaylorSeer extrapolates them, and AdaptiveDiffusion uses the flow adaptation. Lumina includes fixed-stride and dense-head controls. Seed matching follows the recorded runner protocol. The displayed inference outputs are uniformly resized for presentation.

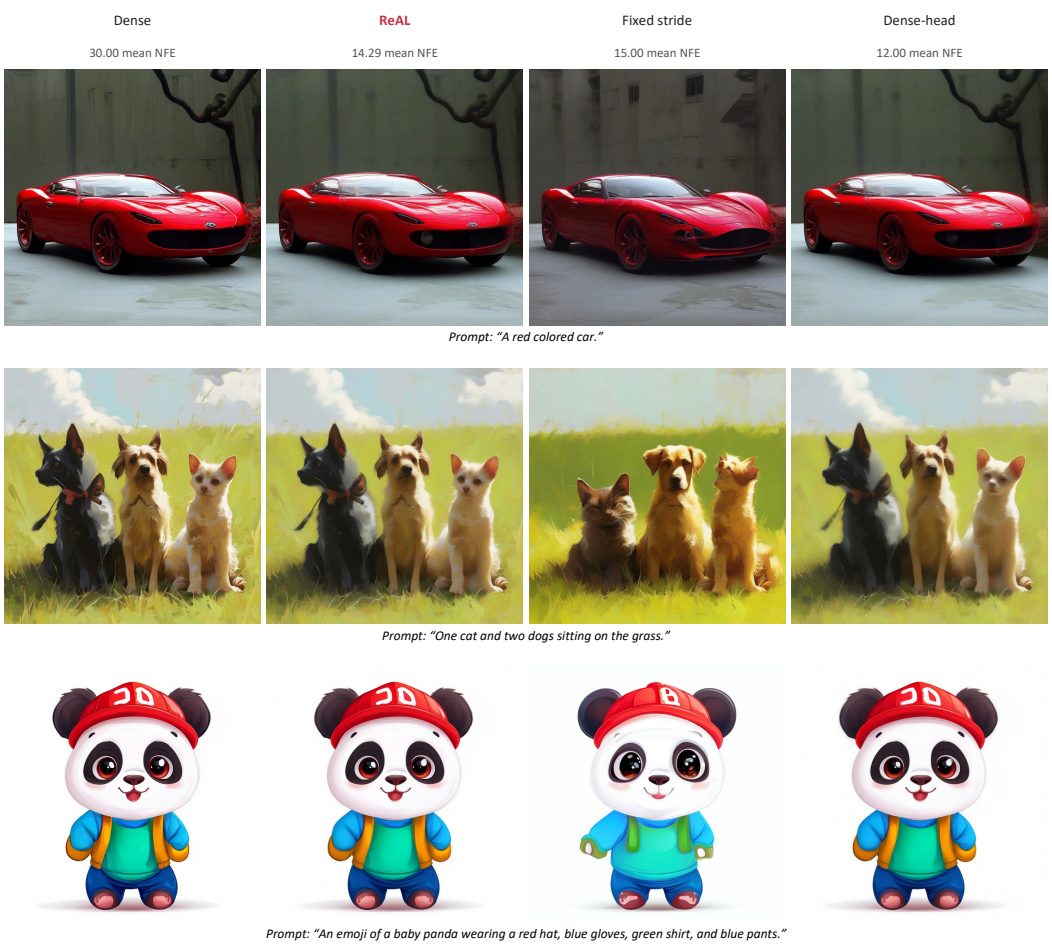

Figure 15: **Lumina-Next-SFT qualitative comparison.** Dense sampling, ReAL, fixed stride, and a dense-head schedule. The fixed-stride and dense-head controls compare allocation strategies on the common prompts.