LoReS: Low-Resolution Selection for Faster-and-Better Diffusion Generation

Anonymous authors
Paper under double-blind review

Abstract

Diffusion inference pursues two opposed goals: making one trajectory cheaper, and spending more to select a better one. What divides them is the price of evidence — selection pays only if a candidate can be judged for far less than it costs to produce — so we move selection into the cheap stage of dynamic-resolution generation. Two measurements license this: at a fixed prompt the initial noise alone moves quality across almost the whole attainable range, for generation and editing alike, and a completed low-resolution trajectory agrees with the full-resolution image its handoff produces in content *and* ranking. LoReS (**Lo**w-**Re**solution noise-trajectory **S**election) batches K low-resolution trajectories, ranks previews with one or several standardized verifiers, and refines a top-T pool rather than the argmax, re-choosing the winner on full-resolution evidence refinement exposes for free. Pooling pays because the cheap ranking is informative and cheap to widen: at $K{=}8$ its top-1 holds the true best for 32.5% of prompts against 12.5% for random, and $T{=}2, 4$ give 53.5% and 76.5%. On FLUX.1-dev/DrawBench200, LoReS is the only training-free accelerator we test that is faster *and* better than the 50-step reference, by a wide margin: at reference latency ImageReward rises 32% with all eight metrics up, where every baseline falls below the reference on ImageReward. LoReS+FreqCa holds seven of eight up at $6.24\times$, pure LoReS six at $9.16\times$, and a $0.260\times$ point lifts all eight; at the quality end $K{=}80$, $T{=}8$ search overtakes Best-of-16 at $2.4\times$ lower latency. Editing repeats on two judges: $7.50 \to 7.70$ at $6.26\times$ (Qwen-Image-Edit), $6.21 \to 6.92$ at $3.52\times$ (FLUX.1-Kontext). LoReS composes with caching, quantization, and distillation; code will be released.

1 Introduction

Diffusion transformers (Peebles & Xie, 2023; Esser et al., 2024) are powerful but expensive iterative generators, and work on efficient inference has developed along two directions that rarely meet. *Single-trajectory acceleration* reduces the cost of producing one sample through fewer steps, distilla-

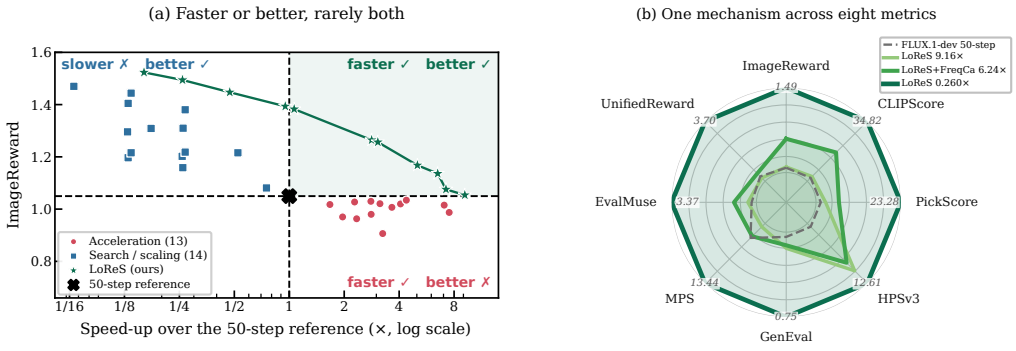

Figure 1: **LoReS is the only training-free method that is both faster and better than its 50-step reference.** (a) speed–quality quadrant. (b) radar, dashed circle = reference; three LoReS points: $9.16\times$ ($6/8$ above), +FreqCa $6.24\times$ ($7/8$), $0.260\times$ ($8/8$).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

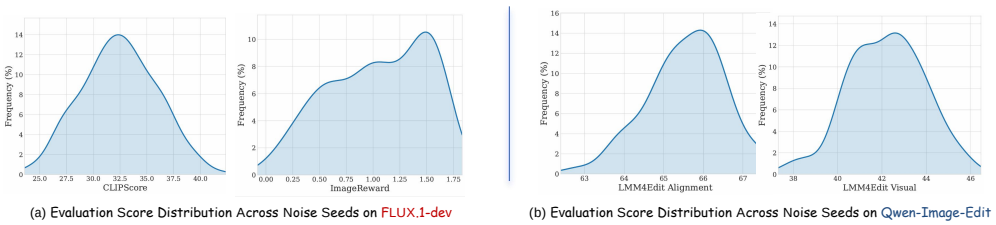

Figure 2: **The initial noise, not the prompt, decides a large part of the outcome.** Score distributions over noise seeds at a fixed prompt set.

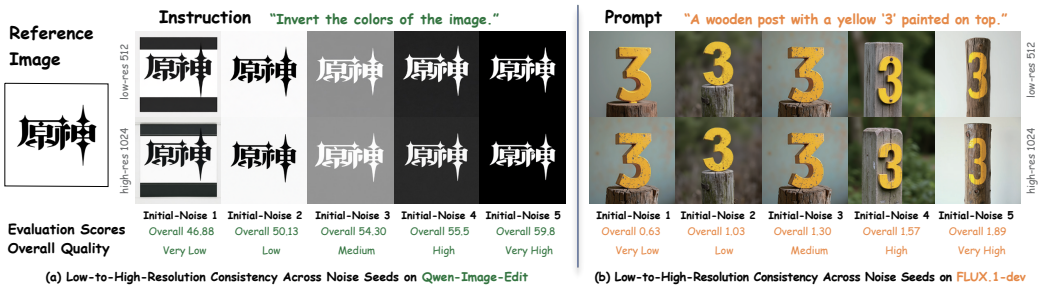

Figure 3: **A 512^2 trajectory already determines what its 1024^2 continuation will look like.** Five noise seeds per panel, each as the low-resolution state and the image the handoff and refinement produce from it. Content and score order both survive the resolution change.

tion, caching, quantization, sparse computation, or dynamic resolution (Liu et al., 2025a;c; Jeong et al., 2026a), accepting whatever trajectory it is handed. *Inference-time scaling* spends additional computation to generate, inspect, or update several candidates before returning one output (Ma et al., 2025a; Singhal et al., 2025), raising the quality ceiling but paying for every candidate at full resolution, which normally makes it slower than a single reference generation. The obstacle is not the search operator but the price of a useful candidate signal.

Observation 1: the seed decides much of the outcome. Fixing the prompt and varying only the initial noise moves quality across nearly the whole attainable range (Fig. 2). With FLUX.1-dev, CLIPScore spreads over roughly 25–41 and ImageReward over 0–1.8, with a left-skewed density, so a randomly drawn seed frequently lands far below what the same model, prompt, and budget could deliver; instruction editing with Qwen-Image-Edit behaves the same way. This unexploited headroom is what any trajectory-selection method converts into quality.

Observation 2: the cheap stage predicts the expensive one. Exploiting that headroom requires knowing which candidate is good before paying for it. Fig. 3 shows that inside a dynamic-resolution sampler this information is already present at 512^2: the low-resolution state and the full-resolution image the handoff produces from it agree on layout, subject, text, and material, and—more importantly for selection—they agree on *order*, across seeds spanning most of the quality range. Scoring the inexpensive stage is therefore not a heuristic proxy for the final image; it is a measurement of the same trajectory that will produce it.

From a winner to a pool. These two observations suggest evaluating candidates where computation is cheap and keeping the evaluated computation. LoReS, **Lo**w-**Re**solution noise-trajectory **S**election, completes K low-resolution trajectories as one batch, decodes and ranks their previews with a configurable verifier, and sends the selected states through the standard resolution handoff (equation 4) into full-resolution refinement. The cheap ranking is already informative, and widening it is cheaper than trusting it: at $K = 8$ the low-resolution verifier puts the true best candidate first for 32.5% of prompts, $2.6\times$ the 12.5% a random draw achieves, and its top 2 and top 4 contain that candidate for 53.5% and 76.5% (Tab. 5a). LoReS therefore promotes a *pool* of the top T candidates, refines all of them, and decides only afterwards, on full-resolution evidence. That final decision is nearly free: the last refinement step of each promoted path already evaluates the velocity field, so its Tweedie

estimate $\hat{x}_0$ is available without an extra transformer call and only needs to be decoded. Because low-resolution steps cost about $1/16$ of full-resolution steps, K can be wide while T stays small, so T dials from committed selection ($T = 1$) to Best-of-K at dynamic-resolution cost.

Results. On DrawBench200 with FLUX.1-dev, one mechanism covers the whole spectrum and is faster *and* better at every point — which no acceleration baseline in our suite reaches, since all fall below the reference on ImageReward. At the reference latency LoReS $(8, 24, 6, 3)$ (configurations are written (L, K, S, T): low-resolution steps, candidate seeds, refinement steps, promoted pool) raises ImageReward from 1.0497 to 1.3827 ($+32\%$) with all eight metrics above the reference; LoReS+FreqCa reaches $6.24\times$ with seven of eight up ($+15\%$ ImageReward), the middle point of Fig. 1; and even at $9.16\times$ six of eight remain above it. Turned toward quality, the $0.260\times$ point improves all eight metrics simultaneously, and LoReS $(18, 80, 12, 8)$ reaches ImageReward 1.5232 — the highest in our evaluation — above Best-of-16 (1.4699) at $2.4\times$ lower latency, while at the reference latency LoReS $(8, 28, 6, 3)$ stays above Best-of-4 at $3.6\times$ lower cost. On instruction editing the gains land on two judges independent of the in-pipeline verifier: overall score rises from 7.50 to 7.70 on Qwen-Image-Edit at $6.26\times$ and from 6.21 to 6.92 on FLUX.1-Kontext at $3.52\times$, while accelerated baselines at comparable speed fall back to or below their own references. LoReS also composes with caches, int8 quantization, and step- or model-distilled backbones.

Our contributions are:

- We connect single-trajectory acceleration and inference-time scaling through the cost of the candidate signal, and measure the two properties that make the connection possible: large seed-induced variance, and consistency of content and ranking across the resolution handoff.
- We propose a training-free method that ranks complete low-resolution trajectories, promotes a top-T pool instead of a single winner, and re-selects on full-resolution Tweedie estimates that its own refinement makes free.
- We evaluate it against broad acceleration and search suites on FLUX.1-dev, on two editing backbones with three judges, and in composition with caches, quantization, and distillation, isolating the role of each knob and of the verifier itself.

2 Related Work

2.1 Efficient single-trajectory generation

One line of work reduces the cost of following a single diffusion trajectory: fast samplers and shortened schedules use fewer denoising evaluations (Song et al., 2021), distillation learns a shorter one (Salimans & Ho, 2022; Black Forest Labs, 2024a), and compressed backbones (Freepik, 2024) or quantization cut per-step arithmetic. Caching and forecasting instead reuse prior activations, predict future features, or selectively update tokens (Ma et al., 2024; Liu et al., 2025a;c; Zou et al., 2025; 2026; Selvaraju et al., 2024; Liu et al., 2025d; 2026), with frequency-aware and calibrated extrapolation as recent variants (Liu et al., 2025b; Zheng et al., 2026b), and sparse attention removes computation inside the attention map (Zhang et al., 2025). Their approximation error becomes visible at aggressive settings, which is why we compare measured quality–latency curves rather than nominal speedups, and why §4.4 composes with them instead of competing against them.

Spatial methods reduce the cost of early structure formation. Cascaded generation allocates computation across resolutions (Ho et al., 2022), while Bottleneck Sampling, RALU, Fresco, and AViTS use bottlenecks, mixed- or progressive-resolution schedules, or adaptive token selection (Tian et al., 2025; Jeong et al., 2026a; Zheng et al., 2026a; Qin et al., 2026). LoReS builds on this dynamic-resolution principle but gives the inexpensive stage a second role: instead of advancing one arbitrary trajectory, it evaluates a batch of low-resolution trajectories and reserves full-resolution refinement for a small promoted set. The $K = 1$ control in §A.1 separates the two effects.

2.2 Inference-time scaling and trajectory selection

Initial noise produces substantial variation even for a fixed prompt (Qi et al., 2024), which Fig. 2 confirms for both generation and editing. Complete-output methods exploit that variation by generating and verifying several images, trading additional full generations for a better returned output (Ma

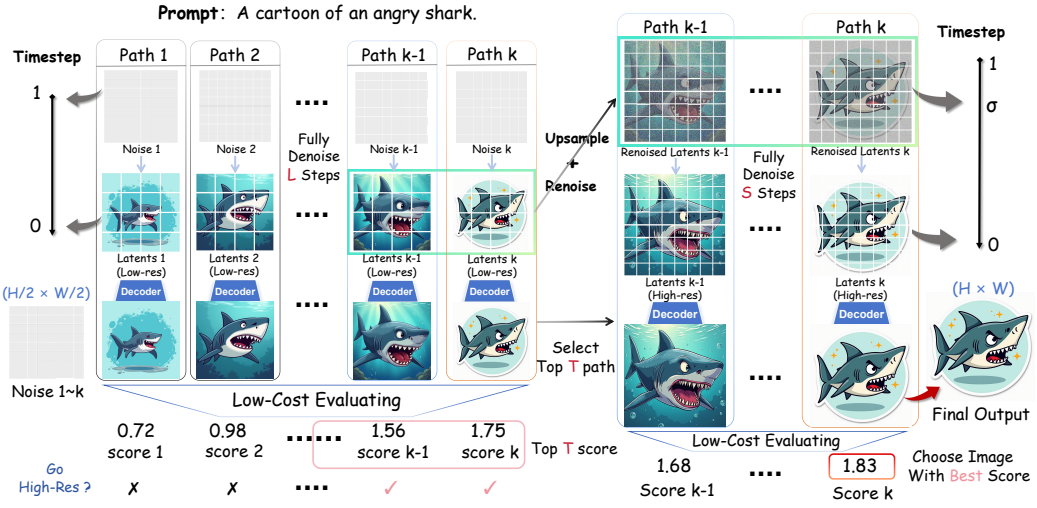

Figure 4: **LoReS** (L, K, S, T)**.** Rank K batched low-resolution trajectories by their decoded previews, promote the top T through the handoff into full-resolution refinement, and return the best of the Tweedie estimates that refinement exposes for free.

et al., 2025a); others optimize or reshape the initial condition through attention, learned perturbations, or frequency priors (Guo et al., 2024; Zhou et al., 2025; Wu et al., 2024; Kim & Kim, 2026). Either way the cost is set by how many full-resolution candidates or optimization steps are needed.

Selection can also happen before trajectories finish. Classical Search branches, prunes, or backtracks over intermediate states (Zhang et al., 2026a); Feynman–Kac steering resamples populations with reward potentials (Singhal et al., 2025); and PSP scores intermediate Tweedie estimates, progressively prunes, and continues survivors (Guimarães & Perona, 2026). PSP is the closest baseline in spirit and strongest probe-based one in Table 1: its probes and survivors live on the full-resolution grid, so every candidate costs full-resolution compute, whereas LoReS probes at $1/16$ the per-step cost and pays full price only for the T promoted paths. LoReS also uses a Tweedie estimate, but to re-rank the promoted set at the end as a free by-product of refinement, not to prune early.

Cheap-probe methods reduce candidate cost with truncated trajectories, skipped layers, attention or activation proxies, previews, or learned latent verifiers (Rawal et al., 2026; Zhang et al., 2026b; Jeong et al., 2026b; Yu et al., 2025; Bucciarelli et al., 2026), and differ in whether the draft is discarded, regenerated, or continued. LoReS evaluates a different object: each probe is a *complete* low-resolution trajectory produced inside a dynamic-resolution sampler, and the selected latents pass through that sampler's own handoff, so the computation spent on scoring also produces the output.

3 Method

3.1 A unified view of inference

For a prompt $\boldsymbol{c}$, let $\mathcal{H}_r$ denote the active trajectory states at search round r. A broad class of inference methods can be written as

$$\mathcal{H}_{r+1} = A_r(\mathcal{H}_r,\, V_r(P_r(\mathcal{H}_r, \boldsymbol{c}))), \qquad \hat{\boldsymbol{x}} = C(\mathcal{H}_R, \boldsymbol{c}), \tag{1}$$

where P_r advances or exposes states for evaluation, V_r scores them, A_r selects or updates the active set, and C completes the surviving state. Single-trajectory acceleration keeps one state and omits selection; complete-output search makes P_r a full generation; iterative search uses A_r to perturb, branch, prune, backtrack, or resample intermediate states. Two quantities decide whether a method can be both selective and fast: the cost of the state that gets scored, and whether the computation spent on the winner is retained by C.

LoReS instantiates equation 1 with two rounds: a batch of complete low-resolution trajectories that keeps a set rather than one element, then full-resolution refinement of every kept state before one

image is returned. Dynamic resolution makes the first round cheap, and the second is cheap to *evaluate* because a full-resolution proxy image falls out of the refinement it already performs.

3.2 LoReS pipeline

A configuration is written LoReS (L, K, S, T): K candidate trajectories, L low-resolution steps, S full-resolution refinement steps, and $T \leq K$ candidates promoted from the first stage to the second. Figure 4 summarizes the four stages. A verifier is any scoring rule on decoded images, and Stage 2 admits one or arbitrarily many of them.

Stage 1: Low-resolution trajectory search. LoReS draws K initial latents from seeds $\{0, \dots, K-1\}$ and completes their L-step schedules on the $(H/f) \times (W/f)$ grid as a single batch, with $f = 2$ throughout, producing clean states $\{\boldsymbol{z}_{\text{lo}}^{(k)}\}_{k=1}^{K}$. These are complete trajectories at their own resolution, not temporal prefixes of full-resolution sampling. Dynamic resolution makes this the inexpensive part of the pipeline and therefore the natural place to create candidate diversity.

Stage 2: Low-resolution ranking and promotion. Each state is decoded to a preview $\boldsymbol{x}_{\text{lo}}^{(k)}$ and scored, and the T highest-ranked indices are promoted. With a single verifier V,

$$s_k = V\left(\boldsymbol{x}_{\text{lo}}^{(k)}, \boldsymbol{c}\right), \qquad \mathcal{T} = \left\{\, k : \; s_k \text{ is among the } T \text{ largest scores} \,\right\}. \tag{2}$$

Nothing restricts the verifier to one scoring rule. For an arbitrary family $V_1, \dots, V_M$ the raw outputs live on incomparable scales, so a weighted sum of raw scores is dominated by whichever metric happens to have the widest spread. We therefore standardize each verifier *within the candidate pool* before combining:

$$s_k = \sum_{m=1}^{M} w_m \, z_m^{(k)}, \qquad z_m^{(k)} = \frac{V_m\left(\boldsymbol{x}_{\text{lo}}^{(k)}, \boldsymbol{c}\right) - \mu_m}{\sigma_m + \epsilon}, \qquad \boldsymbol{w} \in \Delta^{M-1}, \tag{3}$$

where μ_m and σ_m are the mean and standard deviation of V_m over the K previews of the same prompt and ϵ guards a degenerate pool. Pool-wise standardization makes the weights scale-free, and it needs no external calibration set because every prompt supplies its own K samples. Equation equation 2 is the case $M = 1$; our default is $M = 2$ on CLIPScore and ImageReward with $\boldsymbol{w} = (w, 1-w)$, so $w = 0$ is pure ImageReward and $w = 1$ pure CLIPScore, and §A.1 sweeps w. Taking a set instead of the single argmax is the one structural change that separates LoReS from selecting a winner outright, and it is motivated by measurement rather than intuition. A low-resolution verifier is a noisy ordering of the final quality, and its top-T contains the true best far more often than its top-1 does (Tab. 5a), so a small set absorbs most of that noise at a cost that grows only in the refinement term of equation 5.

Stage 3: Handoff and refinement. Every promoted state is carried into full-resolution generation by the same continuation map

$$\tilde{\boldsymbol{z}}^{(k)} = \Phi\left(\boldsymbol{z}_{\text{lo}}^{(k)}\right) = \text{Renoise}_{\rho}\left(\text{Lift}_{\eta}\left(\boldsymbol{z}_{\text{lo}}^{(k)}\right)\right), \qquad k \in \mathcal{T}, \tag{4}$$

where Lift_{η} expands the latent grid by a Hadamard transform blended toward plain replication with weight η, following Fresco (Zheng et al., 2026a), and Renoise_{ρ} is a light SDEdit step (Meng et al., 2022) that re-enters the schedule at progress ρ. We use $\eta = 0.85$ and $\rho = 0.09$ on FLUX.1-dev and re-tune both for editing; App. A.4 lists the remaining settings. Each $\tilde{\boldsymbol{z}}^{(k)}$ is then refined for S full-resolution steps. LoReS therefore reuses a transformed winner state; it does not copy an exact full-resolution prefix and does not restart from a seed under a different sampler.

Stage 4: Full-resolution re-selection. The last refinement step of each promoted path already evaluates the velocity field at $\boldsymbol{x}_t^{(k)}$, so its flow-matching Tweedie estimate $\hat{\boldsymbol{x}}_0^{(k)} = \boldsymbol{x}_t^{(k)} - \sigma_t v_\theta(\boldsymbol{x}_t^{(k)}, t, \boldsymbol{c})$ costs no additional transformer evaluation. Decoding it gives one full-resolution image per promoted candidate for the price of a VAE pass, and the scoring rule of equation 2 or equation 3 is re-applied to these images to return $k^\star = \arg\max_{k \in \mathcal{T}} s_k$; only that path's latent is decoded as the output. The second selection round therefore runs on full-resolution evidence at no denoising cost, which is what makes $T > 1$ affordable.



Figure 5: **Qualitative comparison at matched or higher speed.** LoReS is faster than every baseline shown: 7.22× on (a) FLUX.1-dev (DrawBench), 3.32× on (b) Qwen-Image-Edit (GEdit-Bench).

3.3 Cost, resources, and what the knobs control

Let κ_{lo} and κ_{hi} be the costs of one low- and full-resolution model evaluation, and let c_V be the cost of one decode-and-score. Then

$$\mathrm{Cost}(L,K,S,T) \approx \underbrace{KL\,\kappa_{\mathrm{lo}}}_{\text{search}} + \underbrace{TS\,\kappa_{\mathrm{hi}}}_{\text{refinement}} + \underbrace{(K+T)\,c_V}_{\text{verification}}. \tag{5}$$

Attention-dominated cost gives $\kappa_{\mathrm{lo}} \approx \kappa_{\mathrm{hi}}/f^4$, i.e. about $1/16$ here, so search is roughly sixteen times cheaper per step than refinement and K can be large while T stays small. We still report wall-clock time and write NFE as $(L{\times}K + S{\times}T)$ rather than treating the stages as interchangeable.

The four knobs have distinct roles, which §A.1 measures one at a time. L decides how faithful the preview is to its own continuation; K how much diversity the verifier can choose from; S how much full-resolution refinement the winner receives; T how much of the ranking risk is deferred to full resolution. Only T and S multiply the expensive term, and T interpolates between two known regimes: $T = 1$ is selection with a single committed winner, $T = K$ is Best-of-K executed at dynamic-resolution cost, so a single family reaches both net-speedup and quality-scaling operating points. Because LoReS changes trajectory allocation rather than the denoiser forward, per-step accelerators, quantization, and compressed backbones can be inserted into either stage (§4.4).

4 Experiments

4.1 Experimental setting and compared methods

Protocols. We evaluate FLUX.1-dev (Black Forest Labs, 2024b) at 1024^2 with guidance 3.5 against its 50-step reference sampler on 200 DrawBench prompts (Saharia et al., 2022), reporting ImageReward (IR) (Xu et al., 2023), CLIPScore (Hessel et al., 2021), PickScore (Kirstain et al., 2023), HPSv3 (Ma et al., 2025b), GenEval (Ghosh et al., 2023), MPS (Zhang et al., 2024), EvalMuse (Han et al., 2024), and UnifiedReward (Wang et al., 2025). The Verifier column names the metric LoReS optimizes; the other metrics are held out, and pure-IR, pure-CLIP, and fused protocols are never pooled. For instruction editing we use GEdit-Bench-EN (Liu et al., 2025e) with FLUX.1-Kontext (Black Forest Labs, 2025) and Qwen-Image-Edit (Qwen Team, 2025b), judged by VIEScore (Ku et al., 2024) under Qwen2.5-VL-72B (Qwen Team, 2025a) and Mantis-8B-Idefics2 (Jiang et al., 2024) — semantic consistency (SC), perceptual quality (PQ), overall score (OS) — and by LMM4Edit (Xu et al., 2025), whose mean of visual, alignment, and preservation is also the in-pipeline verifier, so its columns favour it by construction; both VIEScore judges carry the argument.

Efficiency and baselines. Latency is wall-clock time per image after warm-up, relative to each backbone's 50-step reference (24.68 s for FLUX.1-dev, 118.60 s for Qwen-Image-Edit, 52.82 s for FLUX.1-Kontext). For LoReS NFE reads $(L{\times}K + S{\times}T)$, as a batched low-resolution step is not a full-resolution one. We compare against families in §2: sparse attention, token-wise and forecast caching (Liu et al., 2026), dynamic-resolution samplers, adaptive token allocation, compressed or quantized backbones; complete-output Best-of-N (BoN), random and zero-order search (Ma et al., 2025a), adaptive depth-first search (Zhang et al., 2026a), and progressive self-pruning (PSP) (Guimarães & Perona, 2026). Hardware, adaptations, and code are in App. A.4–A.5.

Table 1: **FLUX.1-dev quality–latency spectrum on DrawBench200 — verifier-guided search regime ($\leq 1.00\times$).** The two sub-blocks run different verifiers and are ranked separately: **bold** best, underline second best per column, reference row excluded. Protocol: §4.1, App. A.5.

Method	Family	Verifier	NFE	Acceleration		ImageReward↑	CLIPScore↑	PickScore↑	HPSv3↑	GenEval↑	MPS↑	EvalMuse↑	UnifiedReward↑
				Latency (s)↓	Speedup↑								
FLUX.1-dev	Reference	–	50	24.68	1.00×	1.0497	32.647	22.642	11.3937	0.6823	12.87	3.1405	3.439
BoN ($N = 4$)	Full output	ImageReward	200	94.17	0.262×	1.3096	33.734	23.108	11.7012	0.7103	13.20	3.3033	3.666
BoN ($N = 8$)	Full output	ImageReward	400	189.01	0.131×	1.4047	34.256	23.143	11.7207	0.7204	13.33	3.3322	3.727
BoN ($N = 16$)	Full output	ImageReward	800	377.07	0.066×	1.4699	34.534	23.194	11.8185	0.7325	13.39	3.3670	**3.772**
Random search (2 rounds)	Full output	ImageReward	300	141.00	0.175×	1.3088	33.841	23.143	11.7688	0.7145	13.21	3.2917	3.635
Zero-order (1 round)	Local search	ImageReward	200	95.45	0.259×	1.2020	33.205	23.006	11.6944	0.7046	13.08	3.2218	3.599
Zero-order (2 rounds)	Local search	ImageReward	400	190.35	0.130×	1.2958	33.730	23.106	11.7575	0.7142	13.27	3.2787	3.659
PSP (8→4→2)	Progressive prune	ImageReward	198	91.91	0.269×	1.3803	34.138	23.093	11.6581	0.7322	13.22	3.3174	3.703
PSP (16→8→4)	Progressive prune	ImageReward	396	181.74	0.136×	1.4438	34.370	23.162	11.6700	0.7398	13.28	3.3552	3.725
Adaptive DFS	Tree/backtrack	ImageReward	62.7 avg.	32.80	0.750×	1.0810	33.200	22.950	11.4754	0.7026	12.87	3.2025	3.613
LoReS (18,80,12,8)	Dynamic-res search	ImageReward	(18×80+12×8)	153.97	0.160×	**1.5232**	**34.877**	23.269	12.3105	**0.7548**	13.43	3.3721	3.697
LoReS (14,64,10,6)	Dynamic-res search	ImageReward	(14×64+10×6)	94.79	0.260×	1.4942	34.823	**23.276**	12.6071	0.7512	**13.44**	**3.3733**	3.699
LoReS (10,48,8,4)	Dynamic-res search	ImageReward	(10×48+8×4)	52.31	0.472×	1.4472	34.268	23.196	12.9304	0.7407	13.22	3.3177	3.631
LoReS (8,28,6,3)	Dynamic-res search	ImageReward	(8×28+6×3)	**25.96**	**0.951**×	1.3934	33.825	23.105	**12.9592**	0.7341	13.08	3.3205	3.722
BoN ($N = 4$)	Full output	CLIPScore	200	94.51	0.261×	1.1586	35.125	23.098	11.6133	0.7128	13.11	3.3007	3.666
BoN ($N = 8$)	Full output	CLIPScore	400	189.01	0.131×	1.1967	36.225	23.163	11.5753	0.7323	13.19	3.2866	3.668
PSP (8→4→2)	Progressive prune	CLIPScore	198	91.61	0.269×	1.2181	35.961	23.212	11.5830	0.7301	13.26	3.3127	3.679
PSP (16→8→4)	Progressive prune	CLIPScore	396	181.56	0.136×	1.2157	36.762	23.221	11.5606	0.7378	**13.32**	3.3119	**3.691**
LoReS (18,80,12,8)	Dynamic-res search	CLIPScore	(18×80+12×8)	154.68	0.160×	**1.2519**	**37.778**	**23.244**	12.0097	**0.7498**	**13.32**	**3.3358**	3.666
LoReS (14,64,10,6)	Dynamic-res search	CLIPScore	(14×64+10×6)	97.18	0.254×	1.2340	37.402	23.190	12.3250	0.7462	13.28	3.3230	3.690
LoReS (10,48,8,4)	Dynamic-res search	CLIPScore	(10×48+8×4)	52.66	0.469×	1.1758	36.678	23.112	12.4196	0.7342	13.20	3.3032	3.689
LoReS (8,28,6,3)	Dynamic-res search	CLIPScore	(8×28+6×3)	**26.14**	**0.944**×	1.1557	35.956	23.063	**12.5493**	0.7297	13.04	3.2801	3.612

Table 2: **FLUX.1-dev quality–latency spectrum on DrawBench200 — acceleration regime, three speed tiers.** Baselines use no verifier and appear once per tier; our rows close each tier. **Bold** best, underline second best per column within a tier, reference row excluded. All quality columns are shown for every row; metrics a baseline's source paper does not report are left as "–" rather than imputed. Protocol: §4.1 and App. A.5.

Method	Family	Verifier	NFE	Acceleration		ImageReward↑	CLIPScore↑	PickScore↑	HPSv3↑	GenEval↑	MPS↑	EvalMuse↑	UnifiedReward↑
				Latency (s)↓	Speedup↑								
FLUX.1-dev	Reference	–	50	24.68	1.00×	1.0497	32.647	22.642	11.3937	0.6823	12.87	3.1405	3.439
Tier 1: $1\times <$ *speedup* $\leq 3\times$													
SpargeAttention	Sparse attention	–	50	14.78	1.67×	1.0176	32.452	22.673	11.3442	0.6801	12.64	3.1401	3.419
TaylorSeer ($N = 4$)	Cache	–	50	8.81	2.80×	1.0298	32.315	22.688	11.3798	0.6821	12.67	3.1502	3.430
Bottleneck Sampling	Dynamic resolution	–	30	10.82	2.28×	1.0272	32.242	22.545	11.3221	0.6803	12.54	3.1397	3.431
LoReS (8,24,6,3)	Dynamic-res search	ImageReward	(8×24+6×3)	23.20	1.064×	**1.3827**	33.868	**23.100**	**13.0058**	**0.7221**	**13.12**	**3.3062**	**3.634**
LoReS (6,8,6,2)	Dynamic-res search	ImageReward	(6×8+6×2)	**8.73**	**2.827**×	1.2643	33.582	23.038	12.8738	0.7154	13.03	3.2888	3.590
LoReS (8,24,6,3)	Dynamic-res search	CLIPScore	(8×24+6×3)	23.35	1.057×	1.1355	**35.811**	23.069	12.6675	0.7042	13.06	3.2924	3.605
LoReS (6,8,6,2)	Dynamic-res search	CLIPScore	(6×8+6×2)	8.79	2.807×	1.1004	34.692	23.019	12.4356	0.7002	12.95	3.2553	3.608
Tier 2: $3\times <$ *speedup* $\leq 6\times$													
ToCa ($N = 10$)	Cache	–	50	7.59	3.25×	0.9064	32.285	22.325	11.3018	0.6785	12.51	3.1279	3.266
TaylorSeer ($N = 5$)	Cache	–	50	6.77	3.65×	1.0068	32.570	22.620	11.3412	0.6820	12.54	3.1411	3.382
LinCa ($N = 6$)	Cache	–	50	5.63	4.38×	1.0342	32.629	–	–	–	–	–	–
RALU	Dynamic resolution	–	18	6.07	4.07×	1.0198	32.372	22.510	11.3219	0.6802	12.56	3.1377	3.370
AViTS	Dynamic resolution	–	30	7.86	3.14×	1.0207	32.494	22.580	11.3635	0.6803	12.82	3.1401	3.423
LoReS (8,6,5,2)	Dynamic-res search	ImageReward	(8×6+5×2)	8.08	3.057×	**1.2563**	33.263	22.973	**13.0021**	**0.7147**	**12.99**	**3.2661**	3.569
LoReS (5,4,4,2)	Dynamic-res search	ImageReward	(5×4+4×2)	**4.90**	**5.036**×	1.1678	33.241	22.856	12.5010	0.7099	12.77	3.2190	3.495
LoReS (8,6,5,2)	Dynamic-res search	CLIPScore	(8×6+5×2)	8.12	3.04×	1.1057	**34.770**	**22.994**	12.9980	0.7043	12.98	3.2625	**3.577**
LoReS (5,4,4,2)	Dynamic-res search	CLIPScore	(5×4+4×2)	4.95	4.987×	1.0641	34.179	22.807	12.3527	0.6981	12.75	3.1874	3.483
Tier 3: *speedup* $> 6\times$													
Fresco	Dynamic resolution	–	11	3.28	7.52×	0.9872	31.974	22.303	11.3005	0.6802	12.53	3.1388	3.314
AViTS	Dynamic resolution	–	14	3.49	7.06×	1.0152	32.381	22.337	11.3145	0.6809	12.48	3.1380	3.388
LoReS (6,4,4,1)	Dynamic-res search	ImageReward	(6×4+4×1)	3.79	6.506×	**1.1360**	33.177	22.826	**12.7056**	**0.7049**	12.83	**3.2105**	**3.507**
LoReS (6,4,4,1)	Dynamic-res search	CLIPScore	(6×4+4×1)	3.82	6.456×	1.0480	**33.700**	**22.860**	12.5186	0.6998	**12.87**	3.1721	3.493
LoReS (5,4,4,1)	Dynamic-res search	Fused (w=0.5)	(5×4+4×1)	3.42	7.22×	1.0756	33.379	22.802	12.3827	0.7027	12.70	3.1736	3.459
LoReS (6,2,4,1)	Dynamic-res search	Fused (w=0.5)	(6×2+4×1)	**2.70**	**9.16**×	1.0538	32.700	22.680	12.3385	0.6921	12.73	3.1514	3.432

4.2 TEXT-TO-IMAGE GENERATION: THE FULL QUALITY–LATENCY SPECTRUM

Tables 1 and 2 split the spectrum at the reference latency — the search regime and the acceleration regime in three speed tiers — both populated by the same (L, K, S, T) mechanism.

Search regime. With $V = \text{IR}$, LoReS $(18, 80, 12, 8)$ reaches IR 1.5232 — the highest anywhere in our evaluation — in 153.97 s, above 1.4699 for Best-of-16 at 377.07 s and 1.4438 for PSP $16\to8\to4$ at 181.74 s: faster and better at once than the strongest search and scaling baselines, while also leading or placing second on MPS and EvalMuse. The ordering holds at matched latency (1.4942 near 95 s against 1.3803 and 1.3096) and survives compression to the reference latency, where LoReS $(8, 28, 6, 3)$ obtains 1.3934 in 25.96 s, above Best-of-4 at $3.6\times$ lower cost, and posts the block's best HPSv3. Under $V = \text{CLIP}$ the pattern repeats on that verifier's own metric. Held-out critics stay favourable; the only column a full-output baseline tops in the IR block is UnifiedReward (Best-of-16).

Acceleration regime. Tier 1 carries the central point: at essentially the reference latency LoReS $(8, 24, 6, 3)$ $(1.064\times)$ posts IR 1.3827, CLIPScore 33.868, PickScore 23.100, HPSv3 13.01, GenEval 0.7221, MPS 13.12, EvalMuse 3.306 and UnifiedReward 3.634 — all eight above the reference at once, with ImageReward up 32% — while SpargeAttention, TaylorSeer, and Bottleneck Sampling sit *below* it. This is not a Tier 1 artifact: LoReS stays above the reference on ImageReward, CLIPScore, and GenEval at every speed up to $9.16\times$ (IR 1.2643 at $2.83\times$, 1.2563 at $3.06\times$, 1.1360 at $6.51\times$, 1.0538 at $9.16\times$), whereas *every* acceleration baseline is below the reference on all three at every

Table 3: **Instruction-based editing on GEdit-Bench (EN).** Speed-ups are relative to each backbone's own 50-step baseline (118.60 s Qwen-Image-Edit, 52.82 s FLUX.1-Kontext), which also supplies the baseline scores. **Bold** marks the best value per column within a speed block. LMM4Edit (Qwen3-VL-8B, 0–100) is also the in-pipeline verifier. Protocol: §4.1.

Method	Family	NFE	Acceleration		GEdit-Bench (EN)								
			Latency (s) ↓	Speed ↑	Qwen2.5-VL-72B			Mantis-8B-Idefics2			LMM4Edit		
					SC ↑	PQ ↑	OS ↑	SC ↑	PQ ↑	OS ↑	Vis. ↑	Align. ↑	Pres. ↑	Overall ↑
(a) Qwen-Image-Edit														
Baseline	Reference	50	118.60	1.00×	7.79	7.57	7.50	3.545	5.073	3.503	56.62	64.65	59.01	60.09
BoN (N=8)	Full output	400	954.55	0.12×	7.91	7.56	7.60	3.378	5.079	3.425	**57.44**	66.09	**59.73**	61.08
LoReS (8,28,8,4)	Dynamic-res search	(8×28+8×4)	**184.82**	**0.64**×	**8.09**	**7.62**	**7.82**	**3.461**	**5.629**	**3.746**	57.41	**67.23**	58.85	**61.16**
SpargeAttention	Sparse attention	50	96.42	1.23×	7.81	7.53	7.50	3.541	5.168	3.545	55.92	64.60	57.02	59.18
ToCa ($N = 6$)	Cache	50	71.88	1.65×	7.89	7.46	7.54	3.550	5.036	3.454	55.76	65.54	57.23	59.51
Fresco	Dynamic resolution	30	**44.09**	**2.69**×	7.82	7.51	7.50	**3.627**	5.358	3.673	56.31	65.60	57.24	59.72
LoReS (5,10,5,3)	Dynamic-res search	(5×10+5×3)	62.66	1.89×	**8.12**	**7.55**	**7.78**	3.438	**5.659**	**3.760**	**57.01**	**66.88**	**58.32**	**60.73**
LoReS (5,10,5,2)	Dynamic-res search	(5×10+5×2)	52.15	2.27×	8.07	7.52	7.75	3.406	5.457	3.625	56.68	66.65	58.09	60.47
RALU	Dynamic resolution	18	27.91	4.25×	7.82	7.52	7.51	3.556	5.322	3.618	56.29	65.57	57.13	59.66
TaylorSeer ($N = 6$)	Cache	50	**27.39**	**4.33**×	7.60	7.37	7.30	**3.568**	5.116	3.520	55.75	63.49	57.02	58.75
LoReS (4,5,5,2)	Dynamic-res search	(4×5+5×2)	35.77	3.32×	8.02	**7.56**	**7.74**	3.403	**5.745**	**3.781**	**56.77**	**66.39**	**57.99**	**60.39**
LoReS (3,4,5,2)	Dynamic-res search	(3×4+5×2)	31.88	3.72×	**8.04**	7.54	7.73	3.402	5.626	3.717	56.54	66.38	57.75	60.22
TaylorSeer ($N = 9$)	Cache	50	22.46	5.28×	6.67	6.63	6.31	**3.550**	5.090	3.490	55.62	63.39	56.78	58.60
LoReS (4,3,5,1)	Dynamic-res search	(4×3+5×1)	**18.95**	**6.26**×	**8.02**	**7.52**	**7.70**	3.404	**5.689**	**3.731**	**56.14**	**66.00**	**57.41**	**59.85**
(b) FLUX.1-Kontext														
Baseline	Reference	50	52.82	1.00×	6.48	7.25	6.21	3.46	4.51	3.12	53.34	57.59	57.28	56.07
BoN (N=4)	Full output	200	211.30	0.25×	6.83	7.31	6.51	3.47	4.69	3.27	**54.67**	60.01	**58.12**	57.60
LoReS (10,24,9,4)	Dynamic-res search	(10×24+9×4)	**81.26**	**0.65**×	**7.46**	**7.34**	**7.13**	**3.69**	**5.04**	**3.48**	54.15	**63.16**	56.65	**57.99**
SpargeAttention	Sparse attention	50	48.45	1.09×	6.46	**7.25**	6.21	3.45	4.49	3.10	53.27	57.62	57.23	56.04
RALU	Dynamic resolution	30	24.56	2.15×	7.06	7.20	6.70	3.50	4.55	3.16	**53.42**	57.65	**57.31**	56.13
LoReS (8,8,6,2)	Dynamic-res search	(8×8+6×2)	25.77	2.05×	**7.39**	**7.25**	**7.04**	**3.65**	**5.00**	**3.42**	52.79	**61.73**	55.89	**56.80**
LoReS (6,5,4,2)	Dynamic-res search	(6×5+4×2)	**16.35**	**3.23**×	7.31	7.21	6.97	3.61	4.66	3.27	51.03	60.78	55.07	55.63
Fresco	Dynamic resolution	18	18.66	2.83×	7.01	7.15	6.66	3.49	4.54	3.15	**53.47**	57.66	57.25	56.13
TaylorSeer ($N = 6$)	Cache	50	**14.68**	**3.60**×	6.47	**7.29**	6.17	3.44	4.48	3.09	53.22	57.61	**57.32**	56.04
LoReS (4,4,5,2)	Dynamic-res search	(4×4+5×2)	15.01	3.52×	**7.23**	**7.29**	**6.92**	**3.68**	**4.75**	**3.28**	51.50	**60.93**	56.13	**56.17**

tier (ToCa 0.9064; RALU, Fresco, and AViTS between 0.987 and 1.021). LoReS degrades slowly because the search term $KL\kappa_{\text{lo}}$ shrinks together with the refinement term, so a fast configuration is a smaller search, not a coarser trajectory; baseline ordering is budget-dependent — Fresco slips to IR 0.9872 at 7.52×. Figure 5(a) shows this per image at 7.22×: baselines keep the global layout but drift on the semantic content a low-resolution verifier rejects; App. A.2 shows these outputs beside the reference.

4.3 Instruction-based editing

The pipeline is unchanged apart from the verifier: LMM4Edit scores the low-resolution previews and the promoted full-resolution paths by the mean of visual, alignment, and preservation.

Qwen-Image-Edit. The strongest evidence is that the gain shows up on the two judges *independent* of the LMM4Edit verifier LoReS optimizes. On GEdit-Bench-EN the 50-step reference scores 7.50 (Qwen2.5-VL-72B OS) and 3.503 (Mantis OS); LoReS raises *both* at every speed we report — 7.82 / 3.746 at 0.64× (above full-output Best-of-8 at under half its latency), 7.78 / 3.760 at 1.89×, 7.74 / 3.781 at 3.32×, 7.73 / 3.717 at 3.72×, and still 7.70 / 3.731 at 6.26×. Accelerated baselines cannot hold this: at a *slower* 5.28×, TaylorSeer N=9 collapses to 6.31 / 3.490, below the reference on both independent judges. Because the gain lands on metrics the pipeline never optimizes, it is selection quality, not verifier overfitting. Figure 5(b) shows the mechanism: instructions needing discrete correctness are what a low-resolution verifier can screen, and what the accelerated baselines miss.

FLUX.1-Kontext. The same independence holds on Kontext, where the reference scores 6.21 (Qwen2.5-VL-72B OS) and 3.12 (Mantis OS). Every LoReS row clears both: VL OS climbs to 7.13 at 0.65× (+0.92), holds 6.97 at 3.23×, and is still 6.92 at 3.52× (+0.71), where the nearest baseline RALU reaches only 6.70 at a *slower* 2.15×; Mantis rises to 3.28 against 3.12, and LoReS $(10, 24, 9, 4)$ also beats full-output Best-of-4 (OS 6.51). We stop the Kontext ladder at the 3–4× band rather than pushing a $> 4\times$ cut; perceptual quality moves less than semantic consistency throughout.

Table 4: **LoReS composes with caching, quantization, and step/model distillation.** Each block pairs an accelerator- or backbone-alone baseline with best-ImageReward and best-CLIPScore LoReS settings. Quality columns match Tables 1–2; GenEval uses the official 553-prompt protocol; "–" marks unmeasured cells. Per block, **bold**/underline mark best and second best, reference excluded.

Method	Family	Verifier	NFE	Acceleration		ImageReward↑	CLIPScore↑	PickScore↑	HPSv3↑	GenEval↑	MPS↑	EvalMuse↑	UnifiedReward↑
				Latency (s)↓	Speedup↑								
FLUX.1-dev	Reference	–	50	24.68	1.00×	1.0497	32.647	22.642	11.3937	0.6823	12.87	3.1405	3.439
FreqCa (n=3)	Frequency cache	–	50	8.63	2.86×	0.9976	32.503	22.531	11.5049	–	**12.85**	3.1368	3.436
LoReS(8,4,4,2)+FreqCa	Frequency cache	ImageReward	(8×4+4×2)	**3.96**	**6.24**×	**1.2105**	33.625	**22.786**	**12.1655**	**0.6895**	**12.85**	**3.1919**	**3.453**
LoReS(8,4,4,2)+FreqCa	Frequency cache	CLIPScore	(8×4+4×2)	**3.96**	**6.24**×	1.0661	**34.171**	22.653	11.7258	0.6872	12.58	3.1433	3.426
FLUX.1-dev-int8	Quantization	–	50	17.46	1.41×	0.9506	32.819	22.828	11.2801	0.6752	12.91	3.1481	3.541
LoReS(8,4,4,2)+int8	Quantization	ImageReward	(8×4+4×2)	**6.20**	**3.98**×	**1.2225**	33.426	**22.948**	**13.0323**	**0.7027**	**12.99**	**3.2441**	**3.546**
LoReS(8,4,4,2)+int8	Quantization	CLIPScore	(8×4+4×2)	**6.20**	**3.98**×	1.0509	**34.166**	22.897	12.7612	0.6975	12.85	3.1951	3.493
FLUX.1-schnell	Step distillation	–	8	3.94	6.27×	0.9197	33.951	22.587	10.2610	0.6474	12.36	3.1762	3.527
LoReS(2,4,2,2) on schnell	Step distillation	ImageReward	(2×4+2×2)	**2.55**	**9.70**×	**1.2754**	33.904	**23.017**	**12.2284**	**0.7164**	**12.96**	**3.3651**	**3.615**
LoReS(2,4,2,2) on schnell	Step distillation	CLIPScore	(2×4+2×2)	**2.55**	**9.70**×	1.1167	**35.218**	22.966	11.7189	0.7081	12.91	3.3538	**3.615**
FLUX.1-lite-8B	Model distillation	–	28	7.97	3.10×	0.9283	32.035	**22.697**	11.1506	0.6161	**12.58**	**3.0390**	**3.466**
LoReS(6,6,4,2) on lite-8B	Model distillation	ImageReward	(6×6+4×2)	**5.77**	**4.28**×	**1.0591**	32.440	22.584	**11.5789**	**0.6185**	12.32	2.9883	3.282
LoReS(6,6,4,2) on lite-8B	Model distillation	CLIPScore	(6×6+4×2)	**5.77**	**4.28**×	0.8765	**33.534**	22.593	11.0349	0.6156	12.31	2.9965	3.294

4.4 Composition with caching, quantization, and distillation

LoReS reallocates trajectories and resolution, caches remove per-step work, quantization changes weight arithmetic, and distillation the backbone: complementary, so Table 4 composes them.

Caching. FreqCa alone runs at $2.86\times$ but stays below the reference (IR 0.9976); adding LoReS turns it into the fastest quality gain in the paper: LoReS +FreqCa$(8, 4, 4, 2)$ runs at $6.24\times$ with IR 1.2105 ($+15\%$), CLIPScore 33.625, PickScore 22.786, HPSv3 12.17, EvalMuse 3.192 — seven of eight metrics above the reference at over $6\times$ speed, the middle point of Fig. 1. Table 4 reports the seven DrawBench200 preference metrics plus GenEval as Tables 1–2 for every composition row.

Quantization and distillation. Weight-only int8 composes transparently: LoReS +int8$(8, 4, 4, 2)$ reaches IR 1.2225 at $3.98\times$ against the re-measured int8-alone DrawBench200 baseline (IR 0.9506, $1.41\times$), HPSv3 13.03, MPS 12.99. On step-distilled FLUX.1-schnell (8-step alone: IR 0.9197 at $6.27\times$), LoReS $(2, 4, 2, 2)$ reaches IR 1.2754 at $9.70\times$ — faster and better than the backbone. On lite-8B, LoReS $(6, 6, 4, 2)$ is faster ($4.28\times$ vs. $3.10\times$) and better on ImageReward (1.0591 vs. 0.9283) than the re-measured backbone. In every family the CLIP-selected row moves CLIPScore above the reference while giving back ImageReward, as the verifier sweep in §A.1 predicts.

4.5 Ablations

The knobs (L, K, S, T), the fusion weight w, and the verifier were swept around the default: L and K drive the gain, S saturates early, and $w \approx 0.3$ lifts ImageReward and CLIPScore together; App. A.1 gives curves, hit rates, and transfer. Removing search isolates its share: at $K{=}1$ the pipeline is dynamic resolution alone (IR 0.9360, below the reference), while $K{=}10$ reaches 1.2775. The pool makes the cheap ranking actionable: top-1 holds the true best on 32.5% of prompts vs 12.5% random, and $T{=}2, 4$ give 53.5% and 76.5%. Spending order follows: full-resolution depth saturates earliest (IR 1.2577 at $S{=}4$ against 1.3113 at $S{=}30$), and the first widening of the pool pays most — $T{=}1 \to 2$ buys $+0.0755$ IR for 28% more latency, where $T{=}4 \to 12$ buys $+0.0273$ for a further $2.3\times$.

5 Conclusion

We presented LoReS, a training-free method turning a dynamic-resolution sampler's cheap stage into a search space: a top-T pool of ranked low-resolution trajectories is promoted to full-resolution refinement, the winner chosen on the Tweedie estimates refinement produces. On FLUX.1-dev one mechanism spans the spectrum: at reference latency all eight metrics rise at once, FreqCa holds seven of eight up at $6.24\times$, pure LoReS six at $9.16\times$, and both independent judges improve on two editing backbones at every speed.

The initial noise alone moves quality across the attainable range, and a completed low-resolution trajectory agrees with its continuation in ranking, making a pool, not an argmax, the right object to refine. Quality depends on the verifier, so tables separate objective from held-out metrics, and peak memory grows with K (App. A.7); learned continuation operators and video are next steps.

AI USE STATEMENT

In this work, we used generative AI tools only for the writing of the paper: editing the manuscript to improve its readability, assisting with typesetting and LaTeX formatting, and drafting parts of the text. We have not used generative AI tools for any of the tasks whose disclosure is required, including generating synthetic data sets, developing theoretical models or conceptual frameworks, formulating mathematical claims, providing ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on the research methodology or experiments, implementing the methods, assisting with translation, cleaning or reformatting data sets, supporting qualitative or thematic data analysis, and interpreting results; all of these were carried out by the authors.

We have reviewed all AI-assisted work. Every passage that AI tools helped to draft, polish, or typeset was read and rewritten by the authors, who verified each technical claim, definition, and citation against the underlying experiments and the cited sources before it entered the manuscript. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

LoReS changes inference-time selection rather than model training, but it can still affect which biases are expressed. A preference model used as verifier may favour particular visual styles, cultural depictions, or prompt interpretations, and maximizing over more candidates can amplify those preferences. Lower inference cost can broaden access to beneficial generation tools while also reducing the cost of producing misleading or harmful synthetic media. Deployments should inherit the base model's safeguards, evaluate verifier behaviour across demographic and cultural groups, document synthetic content, and not treat higher reward-model scores as guarantees of human value. Our experiments use public benchmarks and released weights, with no human subjects.

REPRODUCIBILITY STATEMENT

Section 3 defines the low-resolution search, the verifier and top-T promotion rule, the Hadamard/SDEdit handoff, the Tweedie-based re-selection, and the cost accounting. Section 4.1 specifies backbones, prompt sets, metrics, judges, candidate seed pools, and the latency aggregation. Appendix A.4 records hardware, software versions, every fixed hyperparameter, and the order-independence check for the promoted pool; Appendix A.5 documents code provenance and each baseline adaptation; Appendix A.6 states what the reported numbers are and what is released to recompute paired statistics. The release will include generation configurations, adaptation patches, per-prompt metric and timing records, and GenEval metadata.

REFERENCES

Black Forest Labs. FLUX.1-schnell. `https://github.com/black-forest-labs/flux`, 2024a.

Black Forest Labs. FLUX. `https://github.com/black-forest-labs/flux`, 2024b.

Black Forest Labs. FLUX.1 Kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

Davide Bucciarelli, Evelyn Turri, Lorenzo Baraldi, Marcella Cornia, and Rita Cucchiara. Tiny inference-time scaling with latent verifiers. *arXiv preprint arXiv:2603.22492*, 2026.

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, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024.

Freepik. FLUX.1-lite-8B: A distilled version of FLUX.1-dev. `https://huggingface.co/Freepik/flux.1-lite-8B`, 2024.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. GenEval: An object-focused framework for evaluating text-to-image alignment. In *NeurIPS*, 2023.

Rogério Guimarães and Pietro Perona. Inference-time scaling of diffusion models via progressive seed pruning. *arXiv preprint arXiv:2607.21591*, 2026.

Xiefan Guo, Jinlin Liu, Miaomiao Cui, Jiankai Li, Hongyu Yang, and Di Huang. InitNO: Boosting text-to-image diffusion models via initial noise optimization. In *CVPR*, 2024.

Shuhao Han, Haotian Fan, Jiachen Fu, Liang Li, Tao Li, Junhui Cui, Yunqiu Wang, Yang Tai, Jingwei Sun, Chunle Guo, and Chongyi Li. EvalMuse-40k: A reliable and fine-grained benchmark with comprehensive human annotations for text-to-image generation model evaluation. *arXiv preprint arXiv:2412.18150*, 2024.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A reference-free evaluation metric for image captioning. In *EMNLP*, 2021.

Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23, 2022.

Wongi Jeong, Kyungryeol Lee, Hoigi Seo, and Se Young Chun. Training-free mixed-resolution latent upsampling for spatially accelerated diffusion transformers. In *CVPR*, 2026a.

Wongi Jeong, Hoigi Seo, and Se Young Chun. Training-free, perceptually consistent low-resolution previews with high-resolution image for efficient workflows of diffusion models. In *CVPR*, 2026b.

Dongfu Jiang, Xuan He, Huaye Zeng, Cong Wei, Max W. F. Ku, Qian Liu, and Wenhu Chen. MANTIS: Interleaved multi-image instruction tuning. *Transactions on Machine Learning Research*, 2024.

Kwanyoung Kim and Sanghyun Kim. Model already knows the best noise: Bayesian active noise selection via attention in video diffusion model. In *ICLR*, 2026.

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 *NeurIPS*, 2023.

Max Ku, Dongfu Jiang, Cong Wei, Xiang Yue, and Wenhu Chen. VIEScore: Towards explainable metrics for conditional image synthesis evaluation. In *ACL*, 2024.

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 *CVPR*, 2025a.

Jiacheng Liu, Peiliang Cai, Qinming Zhou, Yuqi Lin, Deyang Kong, Benhao Huang, Yupei Pan, Haowen Xu, Chang Zou, Junshu Tang, Shikang Zheng, and Linfeng Zhang. FreqCa: Accelerating diffusion models via frequency-aware caching. *arXiv preprint arXiv:2510.08669*, 2025b.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with TaylorSeers. *arXiv preprint arXiv:2503.06923*, 2025c.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Fei Ren, Shaobo Wang, Kaixin Li, and Linfeng Zhang. SpeCa: Accelerating diffusion transformers with speculative feature caching. In *ACM Multimedia*, 2025d.

Jinshan Liu, Haoran Qin, Xiaobing Tu, Jiacheng Liu, Jiahui Hu, Zhengan Yan, Yukun Xie, Kerui Shen, Jinkui Ren, Yuqi Lin, Xiantao Zhang, and Linfeng Zhang. LinCa: Accelerating diffusion models via learnable decomposed feature caching. In *ECCV*, 2026. arXiv:2608.17973.

Shiyu Liu, Yucheng Han, Peng Xing, Fukun Yin, Rui Wang, Wei Cheng, Jiaqi Liao, Yingming Wang, Honghao Fu, Chunrui Han, et al. Step1X-Edit: A practical framework for general image editing. *arXiv preprint arXiv:2504.17761*, 2025e.

Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, and Saining Xie. Scaling inference time compute for diffusion models. In *CVPR*, 2025a.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *CVPR*, 2024.

Yuhang Ma, Xiaoshi Wu, Keqiang Sun, and Hongsheng Li. HPSv3: Towards wide-spectrum human preference score. In *ICCV*, 2025b.

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 *ICLR*, 2022.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

Zipeng Qi, Lichen Bai, Haoyi Xiong, and Zeke Xie. Not all noises are created equally: Diffusion noise selection and optimization. *arXiv preprint arXiv:2407.14041*, 2024.

Haoran Qin, Zhengan Yan, Shikang Zheng, Xiaobing Tu, Jiacheng Liu, Yuqi Lin, Chang Zou, Jinshan Liu, Peiliang Cai, Xiantao Zhang, Jinkui Ren, and Linfeng Zhang. AViTS: Adaptive spatiotemporal token selection for efficient dynamic-resolution generation. In *ECCV*, 2026. arXiv:2608.17995.

Qwen Team. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025a.

Qwen Team. Qwen-Image technical report. *arXiv preprint arXiv:2508.02324*, 2025b.

Ruchit Rawal, Reza Shirkavand, Sayak Paul, Yuxin Wen, Heng Huang, Yizheng Chen, Tom Goldstein, and Gowthami Somepalli. Flash-BoN: Instant drafts for inference-time scaling in diffusion models. In *ECCV*, 2026. arXiv:2607.04461.

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 *NeurIPS*, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2022.

Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. FORA: Fast-forward caching in diffusion transformer acceleration. *arXiv preprint arXiv:2407.01425*, 2024.

Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen McKeown, and Rajesh Ranganath. A general framework for inference-time scaling and steering of diffusion models. In *ICML*, 2025.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021.

Ye Tian, Xin Xia, Yuxi Ren, Shanchuan Lin, Xing Wang, Xuefeng Xiao, Yunhai Tong, Ling Yang, and Bin Cui. Training-free diffusion acceleration with bottleneck sampling. *arXiv preprint arXiv:2503.18940*, 2025.

Yibin Wang, Yuhang Zang, Hao Li, Cheng Jin, and Jiaqi Wang. Unified reward model for multimodal understanding and generation. *arXiv preprint arXiv:2503.05236*, 2025.

Tianxing Wu, Chenyang Si, Yuming Jiang, Ziqi Huang, and Ziwei Liu. FreeInit: Bridging initialization gap in video diffusion models. In *ECCV*, 2024.

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 *NeurIPS*, 2023.

Zitong Xu, Huiyu Duan, Bingnan Liu, Guangji Ma, Jiarui Wang, Liu Yang, Shiqi Gao, Xiaoyu Wang, Jia Wang, Xiongkuo Min, Guangtao Zhai, and Weisi Lin. LMM4Edit: Benchmarking and evaluating multimodal image editing with LMMs. In *ACM Multimedia*, 2025.

Qingtao Yu, Changlin Song, Minghao Sun, Zhengyang Yu, Vinay Kumar Verma, Soumya Roy, Sumit Negi, Hongdong Li, and Dylan Campbell. TTSnap: Test-time scaling of diffusion models via noise-aware pruning. *arXiv preprint arXiv:2511.22242*, 2025.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. SpargeAttention: Accurate and training-free sparse attention accelerating any model inference. In *ICML*, 2025.

Sixian Zhang, Bohan Wang, Junqiang Wu, Yan Li, Tingting Gao, Di Zhang, and Zhongyuan Wang. Learning multi-dimensional human preference for text-to-image generation. In *CVPR*, 2024.

Xiangcheng Zhang, Haowei Lin, Haotian Ye, James Zou, Jianzhu Ma, Yitao Liang, and Yilun Du. Inference-time scaling of diffusion models through classical search. In *ICLR*, 2026a.

Yunzhe Zhang, Hongfu Liu, and Pengyu Hong. Boosting text-to-image diffusion models via core token attention-based seed selection. *arXiv preprint arXiv:2605.19532*, 2026b.

Shikang Zheng, Guantao Chen, Landis He, Jiacheng Liu, Yuqi Lin, Chang Zou, and Linfeng Zhang. From sketch to fresco: Efficient diffusion transformer with progressive resolution. In *CVPR*, 2026a.

Shikang Zheng, Liang Feng, Xinyu Wang, Qinming Zhou, Peiliang Cai, Chang Zou, Jiacheng Liu, Yuqi Lin, Junjie Chen, Yue Ma, and Linfeng Zhang. Forecast then calibrate: Feature caching as ODE for efficient diffusion transformers. In *AAAI*, 2026b.

Zikai Zhou, Shitong Shao, Lichen Bai, Shufei Zhang, Zhiqiang Xu, Bo Han, and Zeke Xie. Golden noise for diffusion models: A learning framework. In *ICCV*, 2025.

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. In *ICLR*, 2025.

Chang Zou, Shikang Zheng, Evelyn Zhang, Runlin Guo, Haohang Xu, Zhengyi Shi, Conghui He, Xuming Hu, and Linfeng Zhang. Rethinking token-wise feature caching: Accelerating diffusion transformers with dual feature caching. *IEEE Transactions on Image Processing*, 35:6211–6220, 2026.

A Appendix

A.1 Ablations: the four knobs, the fusion weight, and the verifier itself

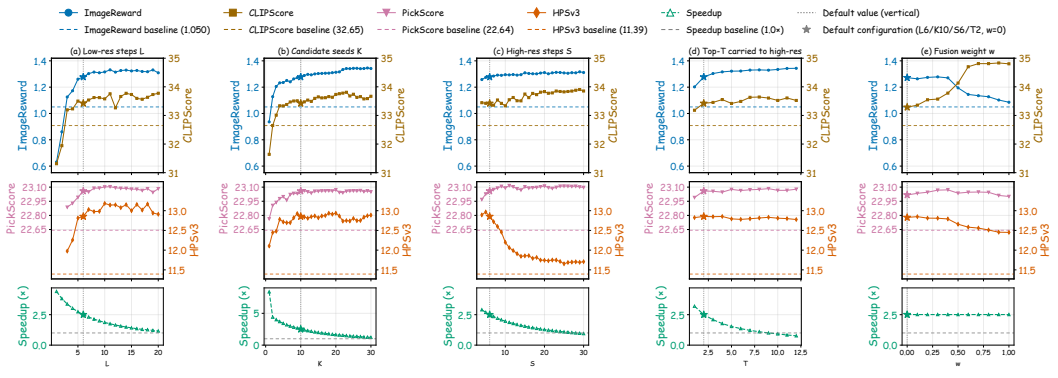

Figure 6: **The four knobs and the fusion weight of LoReS**, each swept around the default configuration (star) on DrawBench200 with FLUX.1-dev; each panel shows what a knob buys and costs.

Table 5: **Low-resolution verifier selection quality at** $K{=}8$ (DrawBench200, FLUX.1-dev). *Hit* is the fraction of prompts whose true best candidate falls inside the top-T; *Score* is the mean of $\max_{k\in\text{top-}T} M_{\text{high}}[k]$. **(a)** fused selector vs. the four selector metrics. **(b)** single-metric selectors (rows) vs. seven oracles at $T{=}2$; **bold** = diagonal, Random = uniform draw at the stated T.

(a) Fused selector vs. retained-set size T

Oracle M		Random (T=1)	T=1	T=2	T=3	T=4	T=5	T=6	T=7	T=8
ImageReward	Hit %	12.5	32.5	53.5	67.0	76.5	86.5	93.0	97.0	100.0
	Score	0.9508	1.2024	1.2899	1.3135	1.3289	1.3397	1.3439	1.3461	1.3473
CLIPScore	Hit %	12.5	27.5	46.0	60.5	69.5	82.5	89.0	94.0	100.0
	Score	32.23	33.65	34.51	34.78	35.01	35.16	35.27	35.37	35.43
PickScore	Hit %	12.5	22.5	37.0	51.0	63.0	75.5	85.0	94.0	100.0
	Score	22.785	23.060	23.221	23.307	23.371	23.405	23.421	23.437	23.450
HPSv3	Hit %	12.5	15.0	29.0	45.0	61.5	73.0	82.0	90.5	100.0
	Score	12.1898	12.6570	13.3216	13.6207	13.8387	13.9304	14.0137	14.0871	14.1339

(b) Hit % at $T{=}2$ (selector × oracle)

Selector	ImageReward	CLIPScore	PickScore	HPSv3	MPS	EvalMuse	UnifiedReward
Random	25.0	25.0	25.0	25.0	25.0	25.0	25.0
ImageReward	**63.0**	38.0	41.0	36.5	39.0	38.5	28.0
CLIPScore	39.0	**50.0**	36.5	28.0	33.5	34.5	26.0
PickScore	33.5	36.0	**53.0**	32.5	39.5	37.0	29.5
HPSv3	33.0	21.5	31.5	**42.0**	34.0	31.5	29.0
MPS	42.0	33.0	43.0	36.5	**55.5**	35.0	27.5
EvalMuse	42.5	33.5	37.0	35.5	38.0	**59.5**	34.5
UnifiedReward	31.5	27.5	28.5	30.0	26.5	32.0	**65.0**

Figure 6 sweeps one knob at a time around $L{=}6$, $K{=}10$, $S{=}6$, $T{=}2$, $w{=}0$ on DrawBench200.

Search is what produces the gain. Panel (b) is decisive. At $K = 1$ the pipeline is dynamic resolution with no selection, scoring IR 0.9360 — below the 50-step reference — at $8.38\times$; widening to $K = 10$, all else fixed, gives 1.2775, and $K = 23$ gives 1.3415. That $+0.34$ cannot come from the resolution schedule, handoff, or refinement budget: none changed. Returns diminish past $K \approx 20$ and memory grows with K, so breadth is a resource decision, not a free parameter.

Where to spend steps. Panels (a) and (c) show that the two step budgets saturate in very different places. Low-resolution depth matters early, IR climbing from 0.6311 at $L = 1$ to 1.2775 at $L = 6$, then flattening near $L \approx 8$–11 while speedup falls from $4.41\times$ to $1.75\times$. Full-resolution steps saturate earlier still, 1.2577 at $S = 4$ against 1.3113 at $S = 30$, because the promoted state already carries the structure. Spending on L and K before S is what enables net-speedup points.

The pool size T. Panel (d) isolates the pooling mechanism. IR rises monotonically with the number of promoted candidates, 1.2020, 1.2775, 1.3035, 1.3160 for $T = 1, \ldots, 4$, while latency grows linearly in T because only the $TS\kappa_{\text{hi}}$ term changes. The first step is the profitable one: $T = 1 \to 2$ buys $+0.0755$ IR for 28% more latency, whereas $T = 4 \to 12$ buys $+0.0273$ for a further $2.3\times$. Table 5a explains the shape: the selector's top 2 and top 4 attain 1.2899 and 1.3289 against the $T = 8$ oracle 1.3473, so a small pool recovers most of it for a small multiple of the refinement cost.

Which verifier, and how much of it. Panel (e) sweeps the fusion weight at fixed latency, as w moves only the objective: IR falls from 1.2721 at $w = 0$ to 1.0854 at $w = 1$ while CLIPScore

rises from 33.288 to 34.804. Curves cross near $w \approx 0.4$–0.5, yet small admixtures are no trade-off: $w = 0.3$ improves both IR (1.2781) and CLIPScore (33.572) over pure IR, and $w \in [0.3, 0.4]$ holds the sweep's best PickScore, while re-measured HPSv3 peaks near $w = 0.1$ (12.85) and falls toward pure CLIP (12.45). Table 5b shows why single verifiers cannot be pooled: each selector recalls its own metric best (42.0–65.0% on the diagonal), while cross-metric transfer is weaker. The verifier is a genuine degree of freedom, deciding which quality axis the pool optimizes.

A.2 Qualitative behavior across the speed ladder

Figure 7 shows images generated by LoReS at 7.22× on FLUX.1-dev, placed beside the 50-step reference on the same four prompts; each LoReS image is the single output its K-candidate pool selected. Layout, rendered text, and fine structure survive at this speed, the qualitative counterpart of the metric ladder in Table 2.



Figure 7: **Images generated by LoReS at** 7.22× **on FLUX.1-dev.** The 50-step reference is placed beside the outputs of LoReS at 7.22×; each LoReS image is the single output its pool selected.

A.3 Inference procedure

Algorithm 1 states the four stages of §3.2 in one place.

Algorithm 1 LoReS inference

Require: prompt $\boldsymbol{c}$; L, K, S, T; verifier V

1: **Stage 1:** $\{\boldsymbol{z}_{\text{lo}}^{(k)}\}_{k=1}^{K} \leftarrow$ batched L-step low-resolution trajectories from seeds $0..K-1$
2: **Stage 2:** $s_k \leftarrow$ score of $\text{VAE}(\boldsymbol{z}_{\text{lo}}^{(k)})$ by equation 2 or equation 3; $\mathcal{T} \leftarrow$ indices of the T largest s_k
3: **for** $k \in \mathcal{T}$ **do**
4: **Stage 3:** $\tilde{\boldsymbol{z}}^{(k)} \leftarrow \Phi(\boldsymbol{z}_{\text{lo}}^{(k)})$; $\boldsymbol{x}^{(k)} \leftarrow \text{Refine}_S(\tilde{\boldsymbol{z}}^{(k)}, \boldsymbol{c})$, caching the Tweedie estimate $\hat{\boldsymbol{x}}_0^{(k)}$
5: **end for**
6: **Stage 4:** $k^\star \leftarrow \arg\max_{k \in \mathcal{T}} s_k$ re-applied to $\text{VAE}(\hat{\boldsymbol{x}}_0^{(k)})$ by equation 2 or equation 3
7: **return** $\text{VAE}(\boldsymbol{x}^{(k^\star)})$

A.4 Implementation and measurement details

LoReS on FLUX.1-dev. All reported configurations use FLUX.1-dev in bfloat16, guidance 3.5, a 512^2 Stage 1 grid ($f = 2$), and a 1024^2 output. Stage 1 runs the complete low-resolution schedule for all K candidates in one batch, with candidate seeds $\{0, \ldots, K-1\}$ reused for every prompt.

The promoted states are lifted by the Hadamard handoff with $\eta = 0.85$, re-noised with SDEdit strength $\rho = 0.09$, and refined for S steps. Verifier scores are candidate-wise standardized before fusion, $w\, z(\mathrm{CLIP}) + (1-w)\, z(\mathrm{IR})$, so $w = 0$ is pure ImageReward and $w = 1$ pure CLIPScore; the single-metric selectors use ImageReward, CLIPScore, PickScore, or HPSv3 unstandardized, since a monotone transform does not change an argmax. Ranking uses a stable sort, so equal scores keep the seed order, and every reported image is fully determined by its configuration.

Promotion and re-selection. The T promoted candidates are refined serially with batch size one, each with its own fixed Stage 2 noise seed derived from its candidate index, so the returned image does not depend on the pool's execution order; we verify this by replaying it in reverse and comparing final latents. The final ranking uses each path's last-step Tweedie estimate $\hat{\boldsymbol{x}}_0 = \boldsymbol{x}_t - \sigma_t v_\theta(\boldsymbol{x}_t, t, \boldsymbol{c})$, which needs no extra transformer evaluation, and only the winner's latent is decoded; the pool therefore adds T VAE decodes and T verifier calls on top of its TS steps.

LoReS on editing backbones. The editing runs keep the same four-stage structure and replace the verifier with LMM4Edit, applied to the low-resolution previews and again to the promoted full-resolution paths, using the mean of visual, alignment, and preservation on every LoReS editing row. They use a 512 Stage 1 grid, a 1024 output, handoff $\eta = 0.8$ and $\rho = 0.1$, and they evaluate the entire GEdit-Bench-EN benchmark (all 303 instructions). Generation runs on seven GPUs with one additional GPU hosting the judge as a resident scoring service.

Runtime environment. Experiments used A100-SXM4-80GB GPUs with PyTorch 2.7.1, CUDA 12.8, a development build of diffusers 0.39, and transformers 4.57. We report the wall-clock times actually measured and never substitute NFE for timing; this is a limitation of cross-repository latency comparison; quality comparisons and within-implementation budget trends are unaffected.

Timing. Every method records synchronized per-prompt wall-clock time for the complete pipeline, including preview decoding, verifier calls, the handoff, and the final decode. For a sharded run we discard the warm-up record, average the last ten records inside each shard, and then average across shards. The FLUX.1-dev reference is 24.6849 s, written 24.68 s; speed is $24.68/\text{latency}$. The Qwen-Image-Edit reference is 118.60 s, measured under the same conditions as the LoReS rows in Table 3(a); non-LoReS accelerators in that block keep previously reported speed-ups with latencies rescaled to the same reference. The FLUX.1-Kontext reference is 52.82 s on the same 303 instructions. For LoReS rows we report NFE as $(L{\times}K + S{\times}T)$ and never add the stages into one number, because a batched low-resolution step costs roughly $1/16$ of a full-resolution one.

Evaluation. DrawBench uses all 200 prompts in their original order, and every selected-image directory contains 200 images with per-image ImageReward, CLIPScore, PickScore, HPSv3, MPS, EvalMuse, and UnifiedReward records where reported. GenEval uses its 553 prompts and the official Mask2Former-based evaluator, and its overall score is the macro average over single object, two objects, counting, colors, position, and color attribution. Table 4 reports the same seven DrawBench200 preference metrics as Tables 1–2. Editing uses the three judges of §4.1, of which two are independent of the in-pipeline verifier and serve as external checks on it.

A.5 Baseline adaptations

PSP. The official release has no FLUX backend, so we ported its predetermined pruning schedule and top-k survivor rule. The FLUX probe is the flow-matching estimate $\hat{\boldsymbol{x}}_{0|t} = \boldsymbol{x}_t - \sigma_t v_\theta(\boldsymbol{x}_t, t, \boldsymbol{c})$, and pruning occurs after steps 12 and 25 of 50, preserving the paper's 25% and 50% positions. The $8{\to}4{\to}2$ schedule has NFE $8(12) + 4(13) + 2(25) = 198$, within 1% of Best-of-4; the larger has 396. Surviving trajectories are continued rather than restarted. We set maximum text length to 256 and use CPU generators so that candidate noise matches the paired LoReS and Best-of-N pools.

Best-of-N and Ma et al. (2025a). The complete-output implementation generates each candidate independently and returns the argmax under the stated verifier. Repeated native Best-of-4 runs measured 94.17 and 94.51 s and produced identical selected images; we use 94.17 s for the pure-ImageReward comparison and 94.51 s for the separately selected CLIPScore row. Random and zero-order search use the public `tt-scale-flux` reproduction, as no official code was released.

Random search with two rounds generates $2 + 4 = 6$ complete images, hence 300 NFE and $141.0\,\mathrm{s}$; zero-order search uses three neighbours at cosine similarity 0.95 per round.

Adaptive depth-first search. This row uses a local FLUX implementation and is not claimed to be the output of an official release. It follows one trajectory on the linear 50-step schedule and triggers SDEdit-style re-noising and backtracking when the intermediate ImageReward criterion $f(\hat{\boldsymbol{x}}_{0|t}) \leq \delta_t$ fires. Across the 200 prompts it averages 1.06 backtracks, each repeating 12 transformer steps, for a mean NFE of $50 + 12(1.06) = 62.72$ and a measured latency of $32.80\,\mathrm{s}$.

Composition implementations. The diffusers FreqCa port applies frequency-domain Hermite extrapolation inside the FLUX transformer and resets its cache state at every LoReS stage boundary; TeaCache and TaylorSeer likewise use separate Stage 1 and Stage 2 cache contexts, so a cache never carries state across the resolution change. Weight-only int8 quantizes the transformer weights only, leaving activations in their native precision. FLUX.1-schnell uses guidance zero and its native 8-step schedule, and lite-8B uses its released compressed checkpoint. Accelerator-alone int8 / schnell (8 steps) / lite-8B rows are re-measured on DrawBench200 and official GenEval under the same evaluation stack as the LoReS rows. Six int8 screening points whose speedup stayed at or below $1.00\times$ were I/O-bound and are therefore excluded from the tables.

Reported and excluded rows. Baseline rows keep the values validated in the cited papers, rescaled to the common reference where needed; metrics those papers do not report are left as "–" rather than imputed. We exclude only runs with incomplete artifacts and unfinalized detector aggregates. Figure 1(a) additionally plots four screening configurations (BoN N=2, ToCa N=6, RALU at $2.34\times$, Fresco at $2.81\times$) that were measured for the spectrum but not promoted into Tables 1–2.

A.6 Statistical reporting

All quality numbers in the paper are point estimates over the full prompt set of the corresponding protocol (DrawBench, GenEval, and GEdit-Bench), and the tables state which protocol each block uses. Because every run stores per-prompt metric records, methods can be joined by prompt index for paired analysis; we release those records together with the generation configurations so that paired t tests, Wilcoxon signed-rank tests, and paired bootstrap intervals can be recomputed on any pair of rows. Comparisons between blocks that use different verifiers are reported as such and not merged into a single ranking.

A.7 Limitations

LoReS needs a low-resolution preview that is informative, memory to hold K batched candidates, and a handoff whose hyperparameters may not transfer unchanged to other samplers. Peak memory grows with K even when latency is held fixed, so breadth is bounded by the device. Selection is verifier-dependent: each verifier most strongly raises its own metric (Tab. 5b), and beyond roughly $5\times$ the pure-LoReS operating points trade off among preference axes: the IR-selected and fused rows keep ImageReward above the 50-step reference at every reported speed, while the CLIP-selected row gives ImageReward back by design. On FLUX.1-Kontext we stop the reported ladder at the 3–$4\times$ band (Table 3b). Cross-repository latency comparisons include those implementation differences.