

Flash-BoN: Instant Drafts for Inference-Time Scaling in Diffusion Models

Ruchit Rawal¹, Reza Shirkavand¹, Sayak Paul², Yuxin Wen¹, Heng Huang¹,
Yizheng Chen¹, Tom Goldstein^{1*}, and Gowthami Somepalli^{1*}

¹ University of Maryland, College Park, MD, USA

² Hugging Face, USA

<https://flash-bon.github.io>

Abstract. Inference-time scaling for text-to-image generation has progressed from simple Best-of- N (BoN) sampling to guided search methods that verify and steer candidate trajectories at intermediate denoising steps. These approaches focus on when and how often to verify during denoising but largely treat the cost of generation itself as fixed. Moreover, the standard practice of comparing methods by number of function evaluations (NFEs) counts only denoising forward passes and ignores verifier overhead, which can distort efficiency rankings. We show that under wall-clock evaluation, simple BoN already matches or outperforms several guided search techniques, suggesting that compute is better spent on broader exploration than on repeated intermediate verification. This motivates **Flash-BoN**, which generates a large pool of inexpensive draft candidates by combining three complementary acceleration knobs: timestep truncation, layer skipping, and activation proxies into a single configuration optimized once per model. An efficient multi-stage verification procedure then identifies the most promising draft, which is refined at full quality. Across three benchmarks and three model scales, **Flash-BoN** consistently outperforms all baselines under fixed wall-clock budgets, with gains that grow at larger model scales (+8% AUC). We further show that our strategy combines well and improves existing orthogonal techniques such as reflection-based prompt optimization (+16% AUC). The gains correlate with increased candidate diversity, which also enables draft-guided selection to accelerate RL post-training convergence.

Keywords: Inference-time scaling · Text-to-image generation · Diffusion models

1 Introduction

Inference-time scaling has become a powerful lever for improving large language models [36], where allocating additional compute at test time enables exploration

Correspondence to: ruchitr@umd.edu.

* Equal advising.

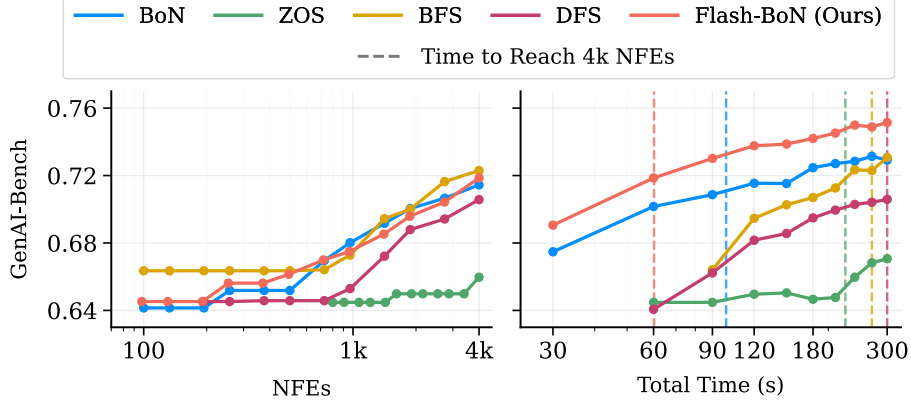


Fig. 1: [T2I Model: Wan2.1 1.3B] Comparison under matched NFE and matched wall-clock budgets. **Left:** At a fixed NFE budget, Breadth-First Search (BFS) is marginally best at high NFEs. **Right:** Under equal runtime, the trend reverses. Dashed vertical lines mark the wall-clock time each method needs to reach 4k NFEs (the maximum NFE shown on the left). Methods with frequent verifier calls take longer to reach the same NFE budget, reducing exploration within a fixed runtime. Consequently, BoN-style approaches outperform under realistic latency constraints.

of multiple solution candidates and selection of the best one. Diffusion-based text-to-image (T2I) models have begun to adopt analogous techniques [21, 26, 43, 45], but existing methods primarily vary *when* and *how often* to verify candidates during intermediate denoising timesteps to assess the potential of ongoing trajectories and prune, guide, or backtrack accordingly. This focus on verification scheduling overlooks an important axis: the cost of generation itself, which we find can be substantially reduced by exploiting redundancies in layer-level computation within and across denoising timesteps.

Inference-time scaling for T2I involves two components: *generation*, which produces candidates via the denoising process, and *verification*, which scores them to select the best output. Best-of- N (BoN) sampling sits at one extreme, investing nearly all compute in generation, producing full trajectories from different seeds and invoking the verifier only once at the end. Guided search techniques sit at the other extreme, invoking the verifier at intermediate timesteps to prune or steer trajectories [26, 43]. This introduces additional design choices such as branching frequency, pruning thresholds, and modifications to the sampling process (e.g., converting deterministic ODE solvers to SDEs). Neither approach questions the per-trajectory generation cost, which raises a natural question: what is the most *efficient way* to allocate a fixed compute budget to verify many potential trajectories and produce the final output for the most promising ones?

Answering this allocation question in practice depends critically on how we measure *compute*. Prior work typically compares inference-time scaling methods using the Number of Function Evaluations (NFEs) as a hardware-agnostic

proxy [21, 26, 43]. However, NFEs only count denoising forward passes and often ignore verifier costs entirely. This matters because the allocation question is fundamentally about *generation versus verification*: if the verifier is unaccounted for, methods that verify more frequently appear efficient even when they are not, especially since the verifier step is often more compute intensive than a single denoising step [18]. In fact, once compute is adjusted to account for *all* forward passes, even simple Best-of- N (BoN) can outperform many sophisticated guided techniques. As illustrated in Fig. 1, BFS appears marginally better under a fixed NFE budget, but this trend reverses under a strict wall-clock budget. Because methods like BFS require frequent intermediate verifier calls, they take substantially longer time to the same NFE limit, sharply reducing the number of candidates they can explore under realistic latency constraints.

This motivates **Flash-BoN**, which combines the strengths of guided search and Best-of- N . Instead of spending the budget on repeated intermediate verification, it generates a large pool of inexpensive *draft* candidates, performs a *single* verification step to select the promising ones, and refines only those at full quality. Our key insight is that several acceleration techniques previously explored in isolation to *reduce generation latency while maintaining quality* can be combined and repurposed for a different objective: *expanding the set of explored candidates under a fixed wall-clock budget*. Concretely, we combine three complementary acceleration knobs: timestep truncation, layer skipping, and activation proxies into a unified draft-generation configuration that jointly determines the speed fidelity trade-off. We formalize this design as a discrete optimization problem over the configuration space and solve it once per model to find the fastest configuration that preserves sufficient fidelity for verifier preferences to transfer after refinement. The resulting drafts are optimized for wall-clock efficiency by design and require no per-prompt tuning at inference time.

Generating a large pool of drafts is only useful if we can reliably identify the one worth refining. Pointwise scoring with an off-the-shelf VLM is efficient but often poorly calibrated, with compressed score ranges that make top candidates hard to distinguish [35, 41]; fully pairwise comparison provides stronger signal but incurs quadratic cost. We introduce a multi-stage selection procedure that combines both: fast pointwise scoring prunes the pool, then targeted pairwise comparisons refine the ranking among a small subset of top candidates, yielding substantially better selections than pointwise alone at a fraction of the pairwise cost.

Together with the multi-stage selection procedure, **Flash-BoN** forms a complete inference-time scaling pipeline. Across three benchmarks and three model scales, it consistently outperforms all baselines. Our main contributions are:

1. **NFE vs. Wall-clock analysis.** NFE comparisons often undercount verifier overhead, distorting method rankings; under wall-clock evaluation, simple BoN already matches or exceeds guided search (Fig. 1).
2. **Draft generation and verification.** We show that different acceleration knobs are complementary and formalize their joint configuration as a discrete optimization problem, yielding a draft policy solved once per model. We also

introduce a multi-stage verification procedure that pairs fast pointwise pruning with targeted pairwise comparisons to reliably identify the best draft.

3. **Consistent gains across scales and benchmarks.** Flash-BoN leads across model–benchmark combinations (Tab. 1), with margins that grow at larger model scales (+8% AUC: area under the score-vs-wall-clock curve).
4. **Modular combination with orthogonal scaling techniques.** The Flash draft strategy functions as a drop-in module that can be layered onto existing methods (such as Reflection-Tuning [45]), improving AUC by 6–16%.
5. **Diversity analysis & RL post-training.** Performance gains correlate with increased candidate diversity (Pearson $r=0.75$), and the draft-guided selection principle transfers to RL: Flash-Flow-GRPO matches baseline convergence in $10\times$ fewer gradient steps.

2 Related Works

Recent work has studied allocating additional compute at inference time [40] to improve diffusion model performance. Broadly, existing approaches either *search* for better trajectories [13, 14, 17, 42] or *steer* sampling using a verifier or reward model without retraining the generator [11, 30]. Ma *et al.* [21] formalize *inference-time scaling* as a search over noise space guided by learned verifiers such as ImageReward [39], and show that increasing compute can yield consistent gains. Singhal *et al.* [26] introduce Feynman–Kac (FK) steering, a particle-based framework that resamples intermediate diffusion states according to reward-defined potentials under deterministic schedulers such as DDIM [27]. Zhang *et al.* [43] connect inference-time scaling to classical tree search, combining local refinement with global exploration via breadth-first search (BFS) or depth-first search (DFS) over denoising trajectories. Other works extend scaling along additional axes: SANA 1.5 [38] studies joint training- and inference-time scaling with best-of- N selection, and ReflectionFlow [45] performs iterative self-refinement through textual feedback.

Despite their differences, these methods share two characteristics: they measure compute primarily via NFEs or FLOPs, abstracting away verifier and control-logic costs; and they use techniques such as timestep truncation or architectural simplifications to reduce latency while preserving baseline quality. In contrast, we treat **wall-clock time** as the primary constraint and repurpose these efficiency techniques for a different goal: expanding the set of explored candidates under a fixed runtime budget by combining complementary draft-generation strategies, jointly optimized once per model without repeated intermediate verification.

3 Problem Setup

Let $\mathcal{D}_\theta$ denote a pre-trained text-to-image diffusion model with parameters θ . Given a text $x \in \mathcal{X}$, standard sampling follows an iterative denoising trajectory through S discrete timesteps $\{t_S, t_{S-1}, \dots, t_1, t_0\}$, starting from Gaussian

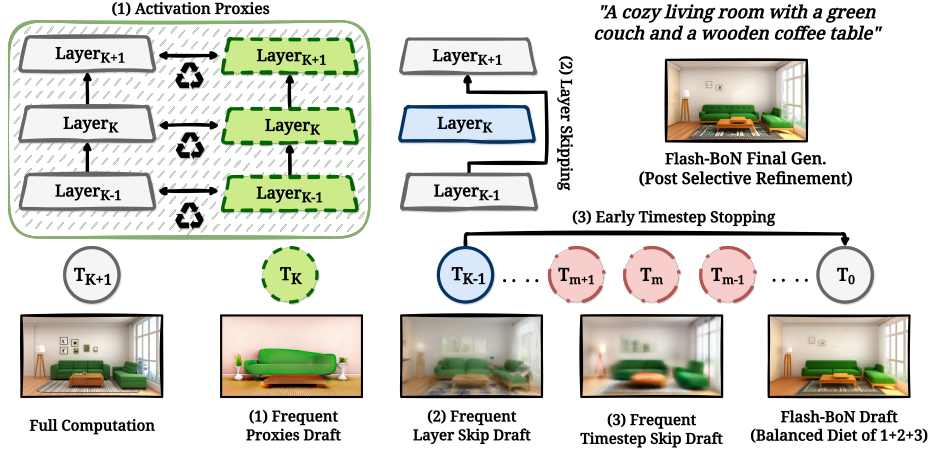


Fig. 2: [T2I Model: Wan2.1 1.3B] Acceleration strategies for draft generation: (1) activation proxies exploit redundancies in activations across timesteps, (2) layer skipping omits redundant layers within a timestep, and (3) early stopping truncates the denoising trajectory. The bottom row compares drafts that over-index on each strategy with our **Flash-BoN** draft (far right), which combines all three through a learned configuration ϕ^* . The balanced **Flash-BoN** draft preserves substantially more structure, enabling reliable verification and refinement.

noise $z_T \sim \mathcal{N}(0, I)$ and progressively denoising to produce the final image: $y = \mathcal{D}_\theta(z_T, x, \{t_i\}_{i=1}^S)$. Each denoising step $z_{t_{i-1}} = f_\theta(z_{t_i}, x, t_i)$ applies a neural network f_θ , typically a U-Net [25] or DiT [24] architecture, consisting of L sequential blocks $\{B_1, B_2, \dots, B_L\}$. While we describe our method in the context of diffusion models for clarity, it applies equally to flow-matching models, which we also evaluate in our experiments.

Given a prompt $x \in \mathcal{X}$ and a wall-clock time budget $T \in \mathbb{R}^+$, our goal is to return an image y^* that maximizes a *verifier* score $V(y, x)$, where V serves as a proxy for a downstream benchmark-specific evaluation metric $E(y, x)$. Inference-time scaling methods therefore generate a pool of candidate images $\{y_j\}_{j=1}^N$ and select

$$y^* = \arg \max_{y \in \{y_j\}_{j=1}^N} V(y, x) \quad \text{s.t.} \quad T_{\text{total}}(\{y_j\}_{j=1}^N, x) \leq T, \quad (1)$$

where T_{total} encompasses all generation, verification, and (when applicable) data reading/loading costs incurred by the method.

4 Flash-BoN

Flash-BoN builds on the observation that promising candidates can often be identified without full-cost generation. Rather than relying solely on early timestep truncation as in prior work, we define a draft configuration ϕ that combines three

acceleration knobs: (i) early timestep stopping, (ii) layer skipping, and (iii) substituting full layer computation with activation proxies. We *precompute* ϕ once per model via discrete optimization to obtain ϕ^* that is fast while preserving sufficient fidelity. **Flash-BoN** operates in three stages:

1. **Draft Generation:** Generate M cheap candidates $\{\tilde{y}_1, \dots, \tilde{y}_M\}$ using ϕ^* .
2. **Verification:** Score all previews with verifier V and select the best draft $\tilde{y}^*$.
3. **Refinement:** Resume generation from $\tilde{y}^*$ with full computation to obtain final output y^* .

4.1 Acceleration Knobs & Precomputing ϕ^*

To generate cheap previews, we must decide which shortcuts to apply without degrading quality. We consider three complementary acceleration techniques:

(1) Early Timestep Stopping. Instead of running all S denoising steps, we generate previews using only the first $S' < S$ steps, inspired by [16, 32]. Let $z_t \in \mathbb{R}^d$ denote the latent at timestep t . The full denoising trajectory is:

$$z_{t_{i-1}} = f_\theta(z_{t_i}, x, t_i) \quad \text{for } i = S, S-1, \dots, 1, \quad (2)$$

where f_θ is the denoising network. Early stopping truncates this to $i = S, \dots, S-S'+1$.

(2) Activation Proxies. We can reduce computation by using *activation proxies* computed from cached features at previously visited timesteps [1, 6, 12]. Since denoising proceeds from high to low timesteps ($t_S \rightarrow t_0$), when processing step i the activations at $t_{i+1}, t_{i+2}, \dots$ are already available. Following Liu *et al.* [18], we forecast the activation $h_{t_i}^{(\ell)}$ for layer ℓ via a second-order Taylor expansion anchored at these cached values. Let $\Delta_t = t_i - t_{i+1}$. We estimate temporal derivatives from cached features:

$$\dot{h}_{t_{i+1}}^{(\ell)} \approx \frac{h_{t_{i+1}}^{(\ell)} - h_{t_{i+2}}^{(\ell)}}{t_{i+1} - t_{i+2}}, \quad (3)$$

and

$$\ddot{h}_{t_{i+1}}^{(\ell)} \approx \frac{h_{t_{i+1}}^{(\ell)} - 2h_{t_{i+2}}^{(\ell)} + h_{t_{i+3}}^{(\ell)}}{(t_{i+1} - t_{i+2})^2}, \quad (4)$$

and then form the activation proxy:

$$\tilde{h}_{t_i}^{(\ell)} = h_{t_{i+1}}^{(\ell)} + \Delta_t \dot{h}_{t_{i+1}}^{(\ell)} + \frac{1}{2} \Delta_t^2 \ddot{h}_{t_{i+1}}^{(\ell)}, \quad (5)$$

which is used as the layer output in place of a full forward computation.

(3) Layer Skipping. While activation proxies exploit redundancy *across timesteps* within a layer, we can also leverage redundancy *across layers* within a single timestep [5, 44]. To do this, we selectively skip a subset of layers $\mathcal{L}_{\text{skip}} \subset \{1, \dots, L\}$ during denoising. At timesteps where skipping is applied, if layer ℓ is skipped, we simply propagate the previous activation:

$$h_{t_i}^{(\ell)} = h_{t_i}^{(\ell-1)} \quad (6)$$

instead of computing the full block output:

$$h_{t_i}^{(\ell)} = B_\ell(h_{t_i}^{(\ell-1)}, x, t_i), \quad (7)$$

where B_ℓ denotes the ℓ -th block of the denoising network.

Configuration Space. The three knobs above define a joint configuration $\phi = (S', n_{\text{skip}}, \ell_{\text{start}}, \ell_{\text{end}}, f_{\text{full}})$, where S' is the number of denoising steps for previews, n_{skip} is the number of initial timesteps before layer skipping begins, $\ell_{\text{start}}, \ell_{\text{end}} \in [1, L]$ specify the contiguous range of layers to skip that is $\mathcal{L}_{\text{skip}} = \{\ell_{\text{start}}, \dots, \ell_{\text{end}}\}$, and f_{full} controls how frequently full (non-proxied) computation is performed. Each ϕ induces a different speed-fidelity trade-off for draft generation; Fig. 2 illustrates how over-indexing on any single knob degrades quality compared to a balanced combination at the same speedup.

Precomputing ϕ^* . We select an optimal configuration ϕ^* once per model by solving a black-box discrete optimization problem that jointly maximizes wall-clock speedup and perceptual similarity (LPIPS) to full-compute outputs, evaluated on a held-out calibration set of 120 prompts. We use dual annealing [29, 37] as the optimizer and select ϕ^* from the resulting Pareto frontier by maximizing speedup subject to a minimum similarity constraint. The optimization runs for 200 iterations; the resulting ϕ^* is then reused across all prompts and benchmarks at inference time. This step is important as the same knobs in a random configuration produce drafts too degraded for the verifier to rank, falling below BoN (App. B.4). Full details of the objective, optimizer comparisons, calibration set design, and the Pareto frontier are provided in Appendix C.

4.2 Flash-BoN Stages

Stage 1: Draft Generation. Using the precomputed configuration ϕ^* , we generate a pool of M preview candidates $\{\tilde{y}_1, \dots, \tilde{y}_M\}$ by sampling different random seeds. Each preview is generated as:

$$\tilde{y}_j = \mathcal{D}_\theta(z_T^{(j)}, x, \phi^*), \quad z_T^{(j)} \sim \mathcal{N}(0, I), \quad (8)$$

where $\mathcal{D}_\theta(\cdot, \cdot, \phi^*)$ denotes the model with acceleration configuration ϕ^* applied.

Stage 2: Verification and Selection. Given M draft candidates $\{\tilde{y}_j\}_{j=1}^M$, the simplest selection strategy is *pointwise scoring*: apply an off-the-shelf verifier V_{pw} independently to each candidate and retain the highest-scoring one,

$$j^* = \arg \max_{j \in [M]} s_j, \quad s_j = V_{\text{pw}}(\tilde{y}_j, x) \in \{1, \dots, 10\}. \quad (9)$$

This is cheap and fully parallelizable, but often unreliable: VLM-based verifiers are noisy and poorly calibrated as absolute scorers [35], and integer-valued outputs compress resolution at the top ranks. In practice, over 80% of prompts produce tied highest scores, with roughly 30% having 20+ candidates sharing the same top score, making selection effectively random (Appendix B.3). *Pairwise* comparison, where the verifier judges two candidates head-to-head, avoids score compression and is better calibrated for relative quality, but incurs prohibitive $O(M^2)$ cost when applied exhaustively.

Multi-stage selection with Elo ranking. We combine both views in a progressive filter that maintains Elo ratings [7] throughout. Let $\rho_k \in (0, 1]$ be the retention ratio at stage k and $\kappa \geq 0$ the average sparse pairwise comparisons per survivor.

1. **Pointwise pruning + Elo seeding.** Score all M candidates and retain the top- ρ_1 fraction. Each candidate’s Elo rating is initialized from its pointwise score via $r_j^{(0)} = 1500 + (s_j - 5) \cdot 100$, seeding subsequent stages with an informed prior:

$$\mathcal{S}_1 = \text{Top}_{\lceil \rho_1 M \rceil}(\{r_j^{(0)}\}_{j=1}^M), \quad M_1 := |\mathcal{S}_1|. \quad (10)$$

2. **Sparse pairwise (adjacent comparisons).** Sort $\mathcal{S}_1$ by Elo and form $C_2 = \lceil \kappa M_1 \rceil$ pairs between *adjacent* entries, concentrating comparisons where the ranking is most uncertain. Each outcome updates both ratings via Elo with a confidence-weighted K -factor.³ Retain the top- ρ_2 fraction:

$$\mathcal{S}_2 = \text{Top}_{\lceil \rho_2 M_1 \rceil}(\{r_j\}_{j \in \mathcal{S}_1}), \quad M_2 := |\mathcal{S}_2|. \quad (11)$$

3. **Dense pairwise tournament.** Run all $\binom{M_2}{2}$ comparisons within $\mathcal{S}_2$, update ratings, and select:

$$\tilde{y}^* = \tilde{y}_{j^*}, \quad j^* = \arg \max_{j \in \mathcal{S}_2} r_j. \quad (12)$$

With our defaults ($\rho_1 = \rho_2 = 0.5$, $\kappa = 1.5$), the procedure uses roughly $\frac{3}{2}M$ verifier calls compared to the $\frac{1}{2}M^2$ required by exhaustive pairwise ranking. We ablate alternative ranking algorithms (Swiss tournaments, knockout brackets, and full pairwise Elo), and effect of varying ρ_1, ρ_2, κ in Appendix B.3.

We distinguish the *verifier* V used at inference time from the *evaluation metric* E used for reporting. The verifier guides candidate selection and is typically an off-the-shelf VLM or reward model, while E denotes benchmark-facing metrics such as programmatic checkers or evaluation-tuned VLMs. Using the same model for both conflates selection with evaluation and risks metric overfitting [21]. We provide qualitative examples for this in Fig. 8.

³ The pairwise verifier returns a confidence level (high/medium/low), scaling K by 1.2/1.0/0.7 respectively.

Stage 3: Selective Refinement. After selecting the best candidate $\tilde{y}^*$, we resume generation from its cached latent state and perform the remaining denoising steps with full computation (i.e., no skipping or proxying). Let S' denote the number of steps used during draft generation. We complete the rest of the trajectory as:

$$y^* = \mathcal{D}_\theta(z_{t_{S'}}^*, x, \phi_{\text{full}}), \quad (13)$$

where $z_{t_{S'}}^*$ is the latent at the draft truncation point and ϕ_{full} corresponds to the standard, unaccelerated configuration. This is valid because the diffusion sampling process is Markovian: future outputs depend only on the current latent, not on how it was produced. The cached state comprises the latent tensor $z_{t_{S'}}^*$, optionally the layer activations $\{h_{t_{S'}}^{(\ell)}\}_{\ell=1}^L$ for warm-starting activation proxies, and the random seed for reproducibility. To limit GPU memory usage during draft generation, we offload cached tensors to CPU and transfer only the selected candidate back for refinement. This data-transfer overhead is absent from NFE accounting but fully captured by our wall-clock evaluation; despite it, **Flash-BoN** still outperforms BoN and BFS in end-to-end runtime.

5 Experiments

We compare **Flash-BoN** against four baselines: Best-of- N (BoN), Breadth-First Search (BFS), Depth-First Search (DFS) [43], and Zeroth-Order Search (ZOS) [21]. Together these cover the main families of inference-time scaling, from simple global exploration to guided trajectory search. For flow-based models (FLUX.1-dev), which use deterministic ODE solvers, we convert to an SDE to enable trajectory branching (Appendix A.1). We evaluate on GenAI-Bench [15] (300 prompts), GenEval [9] (full), and UniGenBench [34] (300 prompts). For GenAI-Bench and UniGenBench, we uniformly subsample from the evaluation set, as running all methods for 300s per prompt on the full benchmarks would be computationally prohibitive (details in Appendix A.2). Experiments span three model scales: Wan 2.1 1.3B, Wan 2.1 14B [31], and FLUX.1-dev [4]. All methods run on NVIDIA H200 GPUs with end-to-end wall-clock timing covering generation, verification, refinement, and data transfer⁴.

Performance under wall-clock constraints. Fig. 3 extends the single-model comparison from Fig. 1 to three architectures spanning an order of magnitude in parameter count: Wan2.1 1.3B, Wan2.1 14B, and FLUX.1-dev. Across all three models, BoN-style approaches that prioritize broad exploration consistently outperform guided search methods that invest compute in intermediate verification. BFS, DFS, and ZOS produce modest gains on the smaller 1.3B model but scale poorly to larger architectures; on Wan2.1 14B and FLUX.1-dev their curves are nearly flat, indicating that repeated verifier calls leave little room for additional candidates as per-image cost grows. **Flash-BoN** improves upon BoN across nearly

⁴ We incorporated warming up of the system to reduce variance.

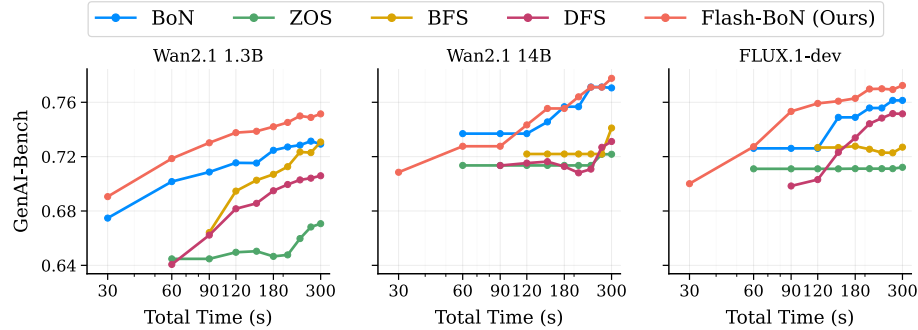


Fig. 3: GenAI-Bench under fixed wall-clock budgets. Performance versus total runtime (seconds). All methods are evaluated under identical hardware and end-to-end time accounting. Best-of- N (BoN) consistently outperforms guided search baselines across models. **Flash-BoN** achieves the strongest performance at all time budgets, with gains that persist and widen at larger scales.

Table 1: Normalized AUC ($\uparrow$) comparison of inference-time search techniques across benchmarks and models. Best results per column are **bold.**

Method	GenAI-Bench			GenEval			UniGenBench		
	Wan 1.3B	Wan 14B	FLUX	Wan 1.3B	Wan 14B	FLUX	Wan 1.3B	Wan 14B	FLUX
ZOS	0.52	0.57	0.57	0.29	0.36	0.37	0.45	0.52	0.49
DFS	0.55	0.50	0.51	0.36	0.37	0.35	0.47	0.46	0.45
BFS	0.50	0.43	0.44	0.31	0.28	0.29	0.43	0.39	0.38
BoN	<u>0.65</u>	<u>0.60</u>	<u>0.60</u>	<u>0.43</u>	<u>0.42</u>	<u>0.42</u>	<u>0.54</u>	<u>0.53</u>	<u>0.54</u>
FLASH-BoN	0.66	0.68	0.68	0.45	0.49	0.47	0.57	0.60	0.61

the entire runtime range, with gains that widen as the budget increases and persist across model sizes. Table 1 summarizes these trends via AUC/Time, the area under the score-versus-wall-clock curve normalized by total time, and shows that **Flash-BoN** leads in every model–benchmark combination across GenAI-Bench, GenEval, and UniGenBench. The advantage over the next best baseline, BoN, is most pronounced on larger models (Wan2.1 14B and FLUX.1-dev), where higher per-image generation cost makes inexpensive drafts particularly effective for expanding exploration within a fixed budget. A practical implication is that Wan2.1 1.3B with **Flash-BoN** at 300s reaches GenAI-Bench scores comparable to Wan2.1 14B at 120–150s, suggesting that inference-time scaling can partially substitute for model size under deployment constraints. Additionally, we break down each knob’s contribution in App. B.4, and show in App. B.5 that the draft-and-select approach still pays off on few-step distilled models like FLUX.1-schnell, where the per-step budget is already small.

Performance across different prompt categories. Fig. 4 breaks down performance on Wan2.1 1.3B by GenAI-Bench prompt category. **Flash-BoN** leads

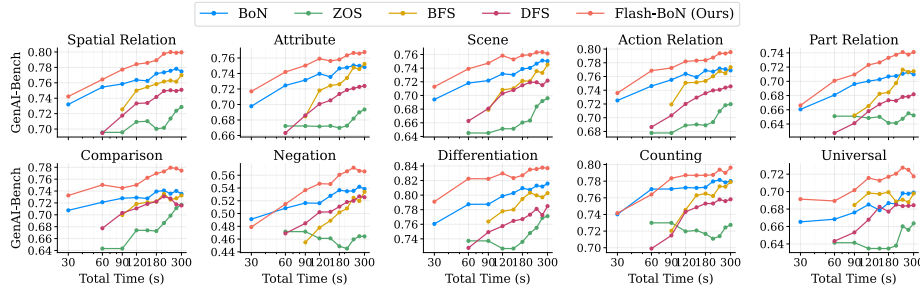


Fig. 4: Per-category GenAI-Bench scores on Wan2.1 1.3B under wall-clock time budgets. **Flash-BoN** leads across all ten categories.

in every category, with the largest gains on compositional prompts (Spatial Relation, Comparison, Part Relation), where the primary failure mode is incorrect layout or missing structure; a broader draft pool increases the chance of sampling a valid composition. The gap narrows for fine-grained categories such as Attribute and Counting, where relevant distinctions are harder to resolve from draft-quality previews. Among guided search methods, BFS is most competitive on Counting and Attribute, where partially formed images already carry informative signal. On holistic categories such as Universal and Negation, where correctness depends on full image context, intermediate verification provides less actionable information and the overhead dominates. Fig. 5 compares the best candidates selected by each method across a range of prompts.

Combining Flash drafts with existing scaling methods. The draft strategy can be layered onto existing inference-time scaling algorithms (Fig. 6). We demonstrate this on two representative methods: Reflection-Tuning [45] and BFS [43]. Reflection-Tuning iteratively generates an image, critiques it via a VLM, and revises the text prompt accordingly; each cycle requires a full image generation. In **Flash-Reflection-Tuning**, the critique loop operates on draft images instead, and a full-quality generation is produced only after the prompt converges, substantially reducing the cost per cycle. BFS prunes trajectories via intermediate verification; in **Flash-BFS**, we accelerate the steps between verification points using drafts, reverting to full computation only after the final pruning decision (see Appendix A.3 for implementation details). To summarize performance across all time budgets in a single number, we report AUC/Time, the area under the score-versus-wall-clock curve normalized by total time.

Flash-Reflection-Tuning improves AUC from 0.46 to 0.62, a large gain because cheaper cycles enable more prompt revisions within the same budget. **Flash-BFS** improves more modestly, from 0.49 to 0.55: faster intermediate steps allow more candidates, but BFS searches locally and cannot escape to distant regions of noise space where compositionally correct solutions may lie. The contrast is instructive: methods that expand the effective search space benefit most

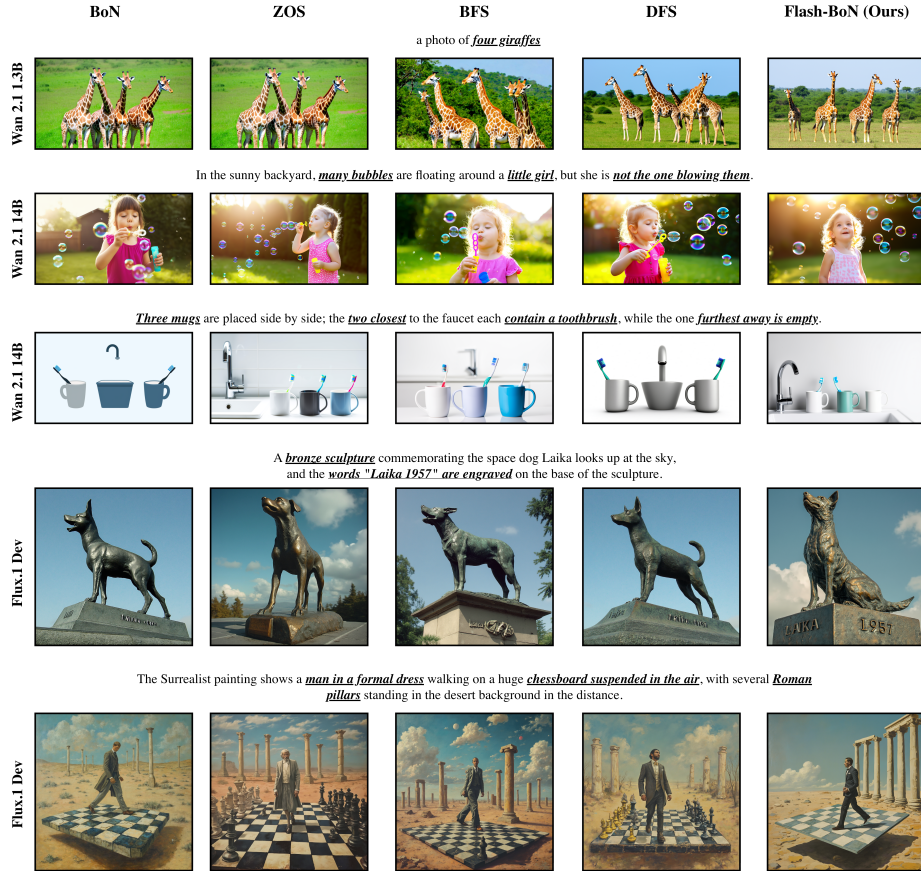


Fig. 5: Qualitative comparison across methods. Each row shows the best image selected by each technique under a fixed 300 s wall-clock budget. Prompts span spatial relations, negation, counting, and complex scene composition. **Flash-BoN** consistently produces images that more faithfully satisfy the prompt.

from cheaper drafts, while those that refine within a fixed region see diminishing returns.

Why does broader exploration help? A diversity analysis. The results above show that exploring more seeds under a fixed budget consistently improves quality. A natural question is whether this simply reflects more lottery tickets or whether effective methods cover a genuinely more *diverse* region of the output space. To examine this, we measure the diversity of each method’s candidate pool using the *Vendi Score* [8], defined as $VS(K) = \exp(-\sum_i \hat{\lambda}_i \log \hat{\lambda}_i)$ with $\hat{\lambda}_i = \lambda_i / \text{tr}(K)$, where K is a pairwise similarity matrix computed from DI-NOv2 [23] features. Intuitively, VS equals 1 for n identical images and n for n perfectly distinct ones. We compute it over all candidates generated within the 300 s

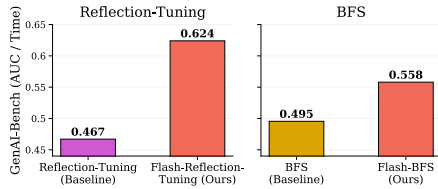


Fig. 6: Combining the Flash draft strategy with existing inference-time scaling methods on Wan2.1 1.3B (GenAI-Bench). We report AUC/Time, the area under the score-vs-wall-clock curve normalized by total time, as a summary efficiency metric. Applying **Flash** drafts to Reflection-Tuning yields a large improvement (0.46 to 0.62), while the gain on BFS is more modest (0.49 to 0.55).

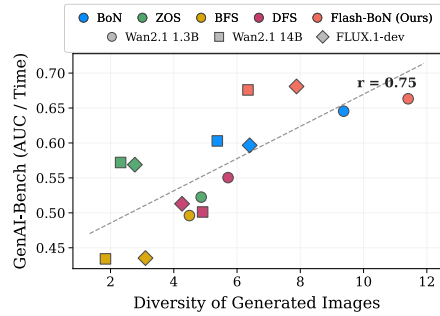


Fig. 7: Candidate pool diversity vs. final GenAI-Bench performance across three models. Diversity is measured using the Vendi Score. Each point is one method averaged over all prompts; color indicates technique, shape indicates model. Dashed line shows the linear correlation fit. Methods that explore more diverse candidates consistently achieve higher quality (Pearson $r=0.75$).

budget for each method–model–prompt combination. As shown in Fig. 7, diversity and final quality are positively correlated (Pearson $r=0.75$). Guided search methods cluster in the low-diversity, lower-performance region, while **Flash-BoN** consistently sits at the high-diversity end, suggesting that inexpensive drafts enable not only more candidates but also *meaningfully different* ones (qualitative examples in Appendix D.3).

Effect of verifier and evaluation metric. Next, we examine how inference-time scaling interacts with the choice of verifier and eval. metric. For each verifier V and metric E , we report the percent improvement of **Flash-BoN** over the no-scaling baseline (Fig. 9). Gains are largest along the diagonal, where verifier matches evaluator, but the magnitude varies strikingly: ImageReward shows 270% improvement versus 40% for HPSv3 and 26% for VQAScore. We hypothesize that ImageReward is particularly susceptible to reward over-optimization: its no-scaling baseline is low relative to the post-scaling score, yet once even a small candidate pool is available, scores plateau almost immediately and remain flat (Appendix B.2), suggesting the model locks onto a narrow preference and stops discriminating. This is consistent with findings that holistic preference models tend to favor aesthetic outputs even when they deviate from the prompt [10, 39]. We share qualitative examples of this failure mode in Appendix D.1.

More broadly, no evaluation metric is bias-free: learned preference models are constrained by their training data and annotation protocols [22, 39], and optimizing against any single metric risks exploiting its blind spots, producing

Qwen2.5 VL Verifier	✗	✗	✓	✓
Eval. Metric (GenEval)	✓	✓	✗	✗
				
	<i>A photo of a black train</i>	<i>A photo of a kite</i>	<i>A photo of a backpack right of a sandwich</i>	<i>A photo of three pizzas</i>

Fig. 8: Failure modes when verifier and evaluator coincide (Wan 2.1 1.3B). Left two columns: the evaluator incorrectly accepts flawed outputs (false positives). Right two columns: the evaluator incorrectly rejects valid outputs (false negatives). When the same model serves both roles, inference-time scaling may exploit these biases, influencing scores without genuine quality improvement.

artifacts not present in the prompt [22]. Fig. 8 illustrates this: when the same model acts as both verifier and evaluator, scaling selects for that model’s biases rather than true quality. A distinct proxy verifier avoids this loop: in our setup, general VLM verifiers produce consistent gains across *all* evaluation metrics (6–14%), even without topping any single matched column, indicating stronger cross-metric generalization. Our multi-stage verification further improves over pointwise scoring across all three metrics, supporting our selection design.

Application to RL post-training. The draft-and-select strategy also applies beyond inference-time scaling. In RL post-training methods such as Flow-GRPO [19], the model generates a group of G rollout trajectories per prompt, scores them with a reward model, and updates the policy via group-relative advantage estimation. Sampling these rollouts is both memory and time intensive, so trajectories are typically drawn at random. For difficult prompts, this makes it unlikely that any rollout in a small group will produce a strong positive example, leaving the optimizer with limited signal. In **Flash-Flow-GRPO**, we replace random sampling with draft-guided selection: generate $2G=16$ inexpensive draft rollouts, score them, and retain the top 6 and bottom 2 candidates for gradient computation, performing full forward passes only on these $G=8$ selected trajectories. The broader draft pool increases the probability of encountering informative positive and negative examples. All hyperparameters are identical to the baseline except for rollout selection and a slightly higher KL penalty to account for the draft-selected distribution (Appendix A.4).

Fig. 10 shows results on Wan2.1 1.3B. **Flash-Flow-GRPO** opens a clear gap within the first 60 steps and maintains it throughout training, reaching 0.699 vs. 0.692 for the baseline. Notably, **Flash-Flow-GRPO** at step 60 already matches the baseline’s final performance, suggesting roughly $10\times$ faster convergence. We show qualitative examples for the same in Appendix D.2.

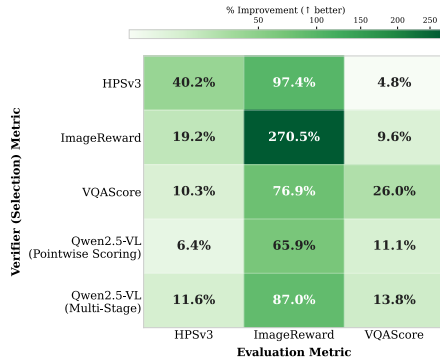


Fig. 9: Percent improvement over no-scaling baseline (Wan 2.1 1.3B). Each cell reports relative improvement when using a given verifier (rows) and evaluating with a specific metric (columns). Diagonal entries, where verifier matches evaluation metric, show the largest gains, while off-diagonal entries reflect cross-metric generalization.

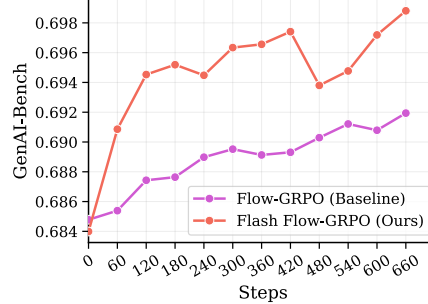


Fig. 10: RL post-training with Flash-BoN rollouts. GenAI-Bench (VQAScore) during Flow-GRPO training on Wan2.1 1.3B. **Flash-Flow-GRPO** converges faster and maintains a consistent advantage throughout training.

6 Discussion

We introduced **Flash-BoN**, an inference-time scaling method that generates a large pool of inexpensive draft candidates, identifies the most promising one via an efficient multi-stage verification procedure, and refines only that candidate at full quality. Across multiple benchmarks and models, **Flash-BoN** consistently outperforms baselines under fixed wall-clock budgets, with gains that grow at larger model scales. These improvements are driven by genuinely broader and more diverse exploration (Pearson $r=0.75$), combine well with orthogonal techniques such as Reflection-Tuning, and transfer to RL post-training, where **Flash-Flow-GRPO** matches baseline convergence while requiring $10\times$ fewer gradient steps. While our results show strong improvements, several open questions remain for future work, such as, extending draft-and-refine scaling to text-to-video with policies that retain temporal consistency and verifiers trained specifically to compare draft-quality images to improve reliability.

Acknowledgments

This work was supported by DOE Office of Science’s ASCR AI for Science initiative, the NSF TRAILS Institute (2229885), and Coefficient Giving, and Longview Philanthropy. R.S. and H.H. were partially supported by NSF IIS 2347592, 2348169, DBI 2405416, CCF 2348306, CNS 2347617, RISE 2536663.

References

1. Agarwal, S., Mitra, S., Chakraborty, S., Karanam, S., Mukherjee, K., Saini, S.K.: Approximate caching for efficiently serving Text-to-Image diffusion models. In: 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24). pp. 1173–1189. USENIX Association, Santa Clara, CA (Apr 2024), <https://www.usenix.org/conference/nsdi24/presentation/agarwal-shubham>
2. Albergo, M., Boffi, N.M., Vanden-Eijnden, E.: Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research* **26**(209), 1–80 (2025), <http://jmlr.org/papers/v26/23-1605.html>
3. Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., Zhong, H., Zhu, Y., Yang, M., Li, Z., Wan, J., Wang, P., Ding, W., Fu, Z., Xu, Y., Ye, J., Zhang, X., Xie, T., Cheng, Z., Zhang, H., Yang, Z., Xu, H., Lin, J.: Qwen2.5-vl technical report (2025), <https://arxiv.org/abs/2502.13923>
4. Black Forest Labs: FLUX.1 [dev]. <https://huggingface.co/black-forest-labs/FLUX.1-dev> (2024), 12 billion parameter rectified flow transformer for text-to-image generation. Accessed: June 28, 2026
5. Chen, G., Zhao, X., Zhou, Y., Qu, X., Chen, T., Cheng, Y.: Towards stabilized and efficient diffusion transformers through long-skip-connections with spectral constraints. In: 2025 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 17708–17718 (2025). <https://doi.org/10.1109/ICCV51701.2025.01645>
6. Chen, P., Shen, M., Ye, P., Cao, J., Tu, C., Bouganis, C.S., Zhao, Y., Chen, T.: δ -dit: A training-free acceleration method tailored for diffusion transformers (2024), <https://arxiv.org/abs/2406.01125>
7. Elo, A.E.: The Rating of Chessplayers, Past and Present. Arco Publishing, New York (1978)
8. Friedman, D., Dieng, A.B.: The vendi score: A diversity evaluation metric for machine learning. *Transactions on Machine Learning Research* (2023), <https://openreview.net/forum?id=g970HbQyk1>
9. Ghosh, D., Hajishirzi, H., Schmidt, L.: Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems* **36**, 52132–52152 (2023)
10. Guo, W.M., Qian, Q., Hasan, K., Du, S.: Aesthetic alignment risks assimilation: How image generation and reward models reinforce beauty bias and ideological" censorship". arXiv preprint arXiv:2512.11883 (2025)
11. Guo, Y., Yang, Y., Yuan, H., Wang, M.: Training-free guidance beyond differentiability: Scalable path steering with tree search in diffusion and flow models. In: The Thirty-ninth Annual Conference on Neural Information Processing Systems (2025), <https://openreview.net/forum?id=nqVLJTPoQC>
12. Habibian, A., Ghodrati, A., Fathima, N., Sautiere, G., Garrepalli, R., Porikli, F., Petersen, J.: Clockwork diffusion: Efficient generation with model-step distillation. In: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8352–8361 (2024). <https://doi.org/10.1109/CVPR52733.2024.00798>
13. He, H., Liang, J., Wang, X., Wan, P., Zhang, D., Gai, K., Pan, L.: Scaling image and video generation via test-time evolutionary search. arXiv preprint arXiv:2505.17618 (2025)
14. Lee, G., Truong, B.N.N., Yoon, J., Lee, D., Kim, M., Bengio, Y., Ahn, S.: Adaptive inference-time scaling via cyclic diffusion search. In: The Thirty-ninth

- Annual Conference on Neural Information Processing Systems (2025), <https://openreview.net/forum?id=jkCH6iwFZm>
15. Li, B., Lin, Z., Pathak, D., Li, J., Fei, Y., Wu, K., Ling, T., Xia, X., Zhang, P., Neubig, G., et al.: Genai-bench: Evaluating and improving compositional text-to-visual generation. arXiv preprint arXiv:2406.13743 (2024)
 16. Li, L., Li, H., Zheng, X., Wu, J., Xiao, X., Wang, R., Zheng, M., Pan, X., Chao, F., Ji, R.: Autodiffusion: Training-free optimization of time steps and architectures for automated diffusion model acceleration. In: 2023 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 7082–7091 (2023). <https://doi.org/10.1109/ICCV51070.2023.00654>
 17. Li, X., Uehara, M., Su, X., Scalia, G., Biancalani, T., Regev, A., Levine, S., Ji, S.: Dynamic search for inference-time alignment in diffusion models. arXiv preprint arXiv:2503.02039 (2025)
 18. Liu, J., Zou, C., Lyu, Y., Chen, J., Zhang, L.: From reusing to forecasting: Accelerating diffusion models with taylorseers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 15853–15863 (October 2025)
 19. Liu, J., Liu, G., Liang, J., Li, Y., Liu, J., Wang, X., Wan, P., ZHANG, D., Ouyang, W.: Flow-grpo: Training flow matching models via online rl. In: The Thirty-ninth Annual Conference on Neural Information Processing Systems (2025), <https://openreview.net/forum?id=oCBKGw5HNf>
 20. Liu, X., Gong, C., Liu, Q.: Flow straight and fast: Learning to generate and transfer data with rectified flow. arXiv preprint arXiv:2209.03003 (2022)
 21. Ma, N., Tong, S., Jia, H., Hu, H., Su, Y.C., Zhang, M., Yang, X., Li, Y., Jaakkola, T., Jia, X., Xie, S.: Scaling inference time compute for diffusion models. In: Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR). pp. 2523–2534 (June 2025)
 22. Ma, Y., Wu, X., Sun, K., Li, H.: Hpsv3: Towards wide-spectrum human preference score. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 15086–15095 (October 2025)
 23. Oquab, M., Darcet, T., Moutakanni, T., Vo, H.V., Szafraniec, M., Khalidov, V., Fernandez, P., HAZIZA, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.Y., Li, S.W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., Bojanowski, P.: DINOv2: Learning robust visual features without supervision. Transactions on Machine Learning Research (2024), <https://openreview.net/forum?id=a68SUt6zFt>, featured Certification
 24. Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 4195–4205 (2023)
 25. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
 26. Singhal, R., Horvitz, Z., Teehan, R., Ren, M., Yu, Z., McKeown, K., Ranganath, R.: A general framework for inference-time scaling and steering of diffusion models. In: Forty-second International Conference on Machine Learning (2025), <https://openreview.net/forum?id=Jp988ELppQ>
 27. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: International Conference on Learning Representations (2021), <https://openreview.net/forum?id=St1giarCHLP>

28. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. In: International Conference on Learning Representations (2021), <https://openreview.net/forum?id=PxtTIG12RRHS>
29. Tsallis, C.: Possible Generalization of Boltzmann-Gibbs Statistics. *J. Statist. Phys.* **52**, 479–487 (1988). <https://doi.org/10.1007/BF01016429>
30. Uehara, M., Zhao, Y., Wang, C., Li, X., Regev, A., Levine, S., Biancalani, T.: Inference-time alignment in diffusion models with reward-guided generation: Tutorial and review. arXiv preprint arXiv:2501.09685 (2025)
31. Wan, T., Wang, A., Ai, B., Wen, B., Mao, C., Xie, C.W., Chen, D., Yu, F., Zhao, H., Yang, J., Zeng, J., Wang, J., Zhang, J., Zhou, J., Wang, J., Chen, J., Zhu, K., Zhao, K., Yan, K., Huang, L., Feng, M., Zhang, N., Li, P., Wu, P., Chu, R., Feng, R., Zhang, S., Sun, S., Fang, T., Wang, T., Gui, T., Weng, T., Shen, T., Lin, W., Wang, W., Wang, W., Zhou, W., Wang, W., Shen, W., Yu, W., Shi, X., Huang, X., Xu, X., Kou, Y., Lv, Y., Li, Y., Liu, Y., Wang, Y., Zhang, Y., Huang, Y., Li, Y., Wu, Y., Liu, Y., Pan, Y., Zheng, Y., Hong, Y., Shi, Y., Feng, Y., Jiang, Z., Han, Z., Wu, Z.F., Liu, Z.: Wan: Open and advanced large-scale video generative models (2025), <https://arxiv.org/abs/2503.20314>
32. Wang, G., Cai, Y., Lijiang Li, Peng, W., Su, S.Z.: PFDiff: Training-free acceleration of diffusion models combining past and future scores. In: The Thirteenth International Conference on Learning Representations (2025), <https://openreview.net/forum?id=wmmDvZGFK7>
33. Wang, W., Gao, Z., Gu, L., Pu, H., Cui, L., Wei, X., Liu, Z., Jing, L., Ye, S., Shao, J., et al.: Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. arXiv preprint arXiv:2508.18265 (2025)
34. Wang, Y., Li, Z., Zang, Y., Zhou, Y., Bu, J., Wang, C., Lu, Q., Jin, C., Wang, J.: Pref-grpo: Pairwise preference reward-based grpo for stable text-to-image reinforcement learning. arXiv preprint arXiv:2508.20751 (2025)
35. Whitehouse, C., Wang, T., Yu, P., Li, X., Weston, J.E., Kulikov, I., Saha, S.: J1: Incentivizing thinking in LLM-as-a-judge via reinforcement learning. In: The Fourteenth International Conference on Learning Representations (2026), <https://openreview.net/forum?id=dnJEH16DI1>
36. Wu, Y., Sun, Z., Li, S., Welleck, S., Yang, Y.: Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In: The Thirteenth International Conference on Learning Representations (2025), <https://openreview.net/forum?id=VNckp7JEHn>
37. Xiang, Y., Sun, D., Fan, W., Gong, X.: Generalized simulated annealing algorithm and its application to the thomson model. *Physics Letters A* **233**(3), 216–220 (1997). [https://doi.org/https://doi.org/10.1016/S0375-9601\(97\)00474-X](https://doi.org/https://doi.org/10.1016/S0375-9601(97)00474-X), <https://www.sciencedirect.com/science/article/pii/S037596019700474X>
38. Xie, E., Chen, J., Zhao, Y., YU, J., Zhu, L., Lin, Y., Zhang, Z., Li, M., Chen, J., Cai, H., Liu, B., Zhou, D., Han, S.: SANA 1.5: Efficient scaling of training-time and inference-time compute in linear diffusion transformer. In: Forty-second International Conference on Machine Learning (2025), <https://openreview.net/forum?id=27h0kXzy9e>
39. Xu, J., Liu, X., Wu, Y., Tong, Y., Li, Q., Ding, M., Tang, J., Dong, Y.: Imagereward: learning and evaluating human preferences for text-to-image generation. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. pp. 15903–15935 (2023)

40. Xu, Y., Deng, M., Cheng, X., Tian, Y., Liu, Z., Jaakkola, T.: Restart sampling for improving generative processes. *Advances in Neural Information Processing Systems* **36**, 76806–76838 (2023)
41. Xuan, W., Zeng, Q., Qi, H., Wang, J., Yokoya, N.: Seeing is believing, but how much? a comprehensive analysis of verbalized calibration in vision-language models. In: Christodoulopoulos, C., Chakraborty, T., Rose, C., Peng, V. (eds.) *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. pp. 1408–1450. Association for Computational Linguistics, Suzhou, China (Nov 2025). <https://doi.org/10.18653/v1/2025.emnlp-main.74>, <https://aclanthology.org/2025.emnlp-main.74/>
42. Yoon, J., Cho, H., Baek, D., Bengio, Y., Ahn, S.: Monte carlo tree diffusion for system 2 planning. In: *Forty-second International Conference on Machine Learning* (2025), <https://openreview.net/forum?id=XrCbBdycDc>
43. Zhang, X., Lin, H., Ye, H., Zou, J., Ma, J., Liang, Y., Du, Y.: Inference-time scaling of diffusion models through classical search. In: *The Fourteenth International Conference on Learning Representations* (2026), <https://openreview.net/forum?id=b7Ftp6U78i>
44. Zhu, H., Tang, D., Liu, J., Lu, M., Zheng, J., Peng, J., Li, D., Wang, Y., Jiang, F., Tian, L., Tiwari, S., Sirasao, A., Yong, J., Wang, B., Barsoum, E.: Dip-go: A diffusion pruner via few-step gradient optimization. In: Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., Zhang, C. (eds.) *Advances in Neural Information Processing Systems*. vol. 37, pp. 92581–92604. Curran Associates, Inc. (2024). <https://doi.org/10.52202/079017-2940>, https://proceedings.neurips.cc/paper_files/paper/2024/file/a845fdc3f87751710218718adb634fe7-Paper-Conference.pdf
45. Zhuo, L., Zhao, L., Paul, S., Liao, Y., Zhang, R., Xin, Y., Gao, P., Elhoseiny, M., Li, H.: From reflection to perfection: Scaling inference-time optimization for text-to-image diffusion models via reflection tuning. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. pp. 15329–15339 (October 2025)