THE SHAPE OF SPECULATION: EFFICIENT DRAFT TOPOLOGY FOR SPECULATIVE JACOBI DECODING IN AUTOREGRESSIVE IMAGE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative Jacobi Decoding (SJD) has recently emerged as an efficient way to accelerate autoregressive image generation by refining multiple tokens in parallel. Recent multi-candidate extensions further improve SJD by introducing alternative continuations, yet their efficiency depends critically on how auxiliary candidates are organized. We argue that existing multi-candidate SJD strategies treat additional speculation as uniformly beneficial, overlooking that the contribution of candidate expansions varies substantially along the Jacobi trajectory. To address this issue, we propose FlexJacobi, a training-free framework that rethinks draft topology construction from the perspective of speculative candidate efficiency. Instead of treating all candidate expansions equally, FlexJacobi devises Verification-aware Topology Shaping (VTS), which derives candidate priorities from current draft distributions and distributes a fixed auxiliary-node budget across the Jacobi trajectory to favor more promising continuation slots. To further improve the effectiveness of these continuations, Context-guided Hybrid Drafting (CHD) is further proposed, which exploits predictive-context consistency with previously generated visual tokens to select continuation entry tokens, while reusing cached draft distributions for the continuation tails. Comprehensive experiments show that FlexJacobi consistently improves decoding efficiency while maintaining competitive generation quality, matching or surpassing SJD-PAC with only 22% of its auxiliary nodes.

1 INTRODUCTION

Autoregressive (AR) models have become a powerful paradigm for text-to-image generation by modeling visual tokens through next-token prediction(Ramesh et al., 2021; Liu et al., 2024; Chen et al., 2025). Despite their strong generation capability, inference remains inherently sequential: each visual token requires a separate forward pass conditioned on preceding tokens. For high-resolution generation, this leads to hundreds or even thousands of forward steps and substantial decoding latency. Speculative Jacobi Decoding (SJD) (Teng et al., 2025) reduces this bottleneck by refining multiple tentative tokens in parallel and reusing previous target-model predictions as draft distributions.

Standard SJD, however, verifies only a single tentative trajectory, so a rejection prevents all subsequent tokens on that trajectory from contributing to the accepted sequence. Recent multi-candidate extensions(Kang et al., 2026; Lei et al., 2026) address this limitation by introducing alternative continuations. SJD-PAC (Kang et al., 2026) makes an early exploration to introduce a shallow local candidates to alleviate frequent rejections at the beginning of the Jacobi path, demonstrating the potential value of additional speculation. Despite its effectiveness, we argue that auxiliary candidates can **contribute very differently** depending on where they are introduced: once a region of the Jacobi trajectory is already covered by sufficient alternatives, further expansion largely provides redundant recovery opportunities, whereas other positions may still lack useful continuations. As illustrated in Figure 1, in the local expansion used by SJD-PAC, additional candidates can improve recovery around early positions, but their contribution gradually saturates as alternatives accumulate within the same region. Redistributing the same speculative capacity to different positions can instead provide useful alternatives where verification would otherwise terminate. These observations suggest that simply adding more candidates is insufficient, and that efficient multi-candidate SJD **instead hinges on how the draft topology organizes** limited speculative capacity along the Jacobi trajectory.

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

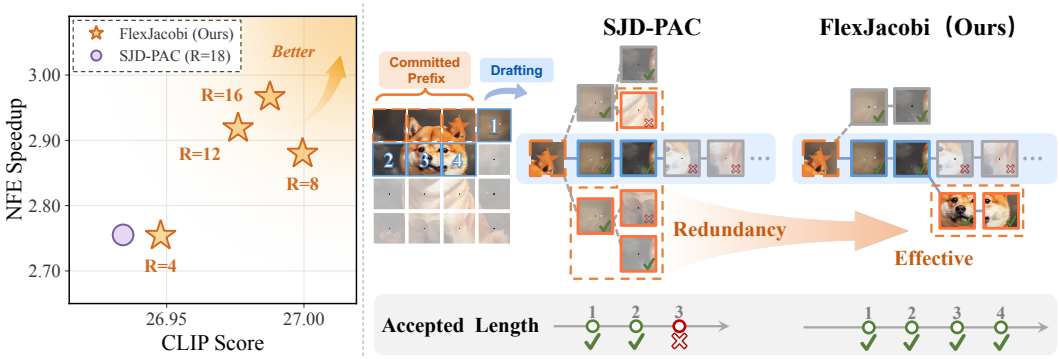


Figure 1: **FlexJacobi against SJD-PAC. Performance(left)**: a better speed–quality trade-off with far fewer auxiliary nodes. **Topology(right)**: SJD-PAC's shallow local topology wastes auxiliary nodes, whereas FlexJacobi's topology effectively extends the accepted sequence.

In this paper, we propose FlexJacobi, a training-free framework for improving auxiliary-candidate efficiency in multi-candidate SJD. FlexJacobi is built on two complementary design principles. First,an auxiliary continuation can contribute only when its preceding trajectory remains sufficiently supported and the current position still admits meaningful alternatives. We therefore design Verification-aware Topology Shaping (VTS), which uses the current draft distributions to prioritize continuation slots and shape a fixed auxiliary-node budget along the Jacobi trajectory. Second, an effective topology alone is insufficient if a continuation starts from an unsuitable token, since a rejected entry prevents the remaining continuation from contributing. Context-guided Hybrid Drafting (CHD) is further proposed, which exploits predictive-context consistency with previously generated visual tokens to select continuation entries. Additionally, since only the entry shares the original Jacobi prefix and thus has a valid cached predictive representation, CHD samples the continuation tail from cached draft distributions without additional target-model forward passes.

Extensive experiments on Lumina-mGPT (Liu et al., 2024) and Janus-Pro (Chen et al., 2025) demonstrate consistent decoding acceleration while maintaining competitive generation quality. It is worth mentioning that, as shown in Figure 1, FlexJacobi with only $R = 4$ auxiliary nodes already surpasses SJD-PAC (Kang et al., 2026) with $R = 18$ nodes in NFE speedup, using only 22% as many auxiliary nodes. FlexJacobi is also complementary to relaxed-verification methods, providing further acceleration when combined with CSD (Wang et al., 2026) and GSD (So et al., 2025).

Our contributions are summarized as follows:

- We propose FlexJacobi, a training-free framework for efficient draft topology design in multi-candidate SJD. Its Verification-aware Topology Shaping (VTS) uses current draft distributions to organize short auxiliary continuations along the Jacobi trajectory under a fixed auxiliary-node budget.
- We introduce Context-guided Hybrid Drafting (CHD), which combines predictive-context-guided entry selection with cached draft-distribution sampling to construct effective auxiliary continuations. This hybrid design improves accepted length without additional model forward passes for drafting.
- We conduct extensive experiments across multiple autoregressive image generators and benchmarks, including controlled budget ablations and compatibility studies with relaxed-verification methods, consistently validating the effectiveness and generality of FlexJacobi.

2 Preliminaries

2.1 Speculative Jacobi Decoding

Given conditioning input c, an autoregressive text-to-image model parameterized by θ generates a sequence of N tokens $x_{1:N} = (x_1, \ldots, x_N)$ from vocabulary $\mathcal{V}$. The generation probability is factorized as $p_\theta(x_{1:N}|c) = \prod_{n=1}^{N} p_\theta(x_n|c, x_{<n})$, where $x_{<n} = (x_1, \ldots, x_{n-1})$. Although autoregressive

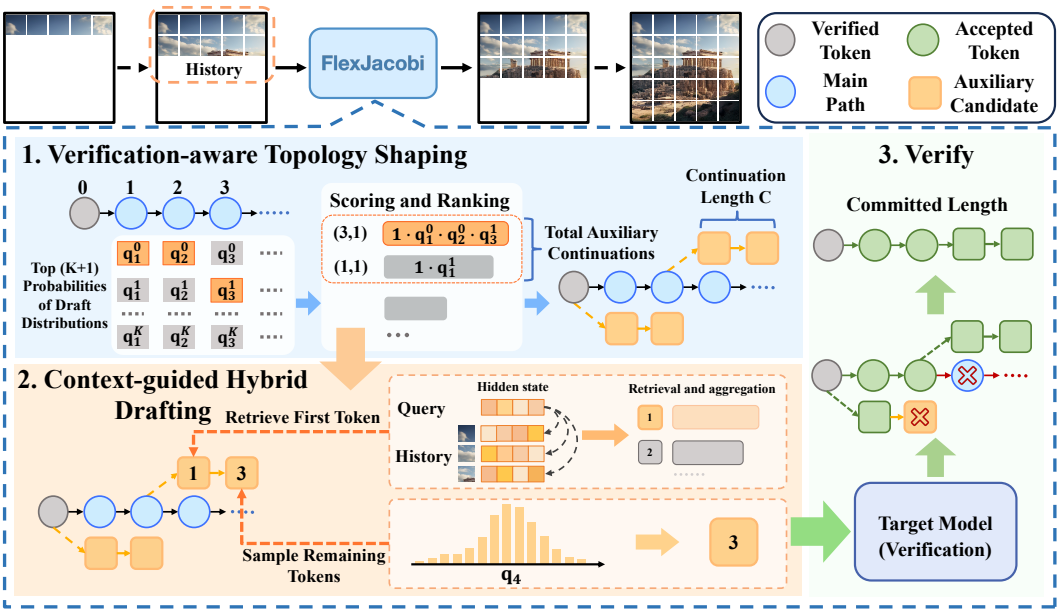


Figure 2: **Overview of FlexJacobi.** *(Step 1)* uses ranked draft probabilities to allocate short auxiliary continuations. *(Step 2)* retrieves entry tokens using hidden-state similarity and randomly samples the remaining tokens from the corresponding draft distributions. *(Step 3)* evaluates all candidates and commits the accepted prefix.

models achieve high-quality image generation, inference requires sequential evaluation of token distributions, resulting in substantial decoding latency for long visual token sequences.

Speculative Jacobi Decoding (SJD) (Teng et al., 2025) accelerates autoregressive image generation by iteratively refining multiple tentative tokens in parallel without introducing an additional draft model.

At each Jacobi iteration r, SJD maintains a tentative trajectory of length L: $Z^{(r)} = (z_1^{(r)}, \ldots, z_L^{(r)})$. Each $z_i^{(r)}$ is generated from its corresponding draft distribution: $z_i^{(r)} \sim q_i^{(r)}(\cdot), \quad i = 1, \ldots, L$, where $q_i^{(r)}$ denotes the proposal distribution at position i. Therefore, the draft distributions $Q^{(r)} = \{q_i^{(r)}\}_{i=1}^{L}$ define the current Jacobi trajectory $Z^{(r)}$.

Given the tentative trajectory, one forward pass of the target model computes the corresponding target distributions:

$$p_i^{(r)}(v) = p_\theta\left(v \mid c, x_{<t}, z_{1:i-1}^{(r)}\right), \quad v \in \mathcal{V},$$

and then the draft proposed token $z_i^{(r)}$ is accepted with probability

$$\alpha_i(z_i^{(r)}) = \min\left(1, \frac{p_i^{(r)}(z_i^{(r)})}{q_i^{(r)}(z_i^{(r)})}\right).$$

When a token fails verification, SJD samples a bonus token from the residual distribution: $\tilde{z}_i \sim \mathrm{Norm}\left([p_i^{(r)} - q_i^{(r)}]_+\right)$, where $[f]_+(\cdot) = \max\{f(\cdot), 0\}$ and Norm normalizes nonnegative values to sum to one(Leviathan et al., 2023; Chen et al., 2023). After verification, the updated predictions are reused to initialize the next Jacobi iteration, producing a new tentative trajectory and corresponding draft distributions.

2.2 Multi-Candidate Drafting in Speculative Jacobi Decoding

In a standard SJD iteration, each position provides **only one tentative token**. Although tokens across positions are refined in parallel, verification still follows a single candidate sequence. Consequently,

once a token fails verification, subsequent tokens in the same trajectory cannot contribute to the accepted sequence, leaving part of the speculative computation unused. To alleviate this limitation, recent works(Lei et al., 2026; Kang et al., 2026) extend SJD with multi-candidate drafting, where additional candidate sequences are introduced to provide alternative verification paths beyond the original Jacobi trajectory. SJD-PAC (Kang et al., 2026) constructs a shallow local candidate tree for the next iteration after rejection. It extends one path into a long chain while the remaining branches terminate within the local tree. This structure provides multiple choices at early positions, whereas later positions on the extended path remain without additional branches. A more detailed discussion of related work is deferred to Appendix A due to space limitations.

Our Motivation. In SJD-PAC (Kang et al., 2026), adding candidates within the local expansion does not necessarily increase accepted length. As illustrated in Figure 1, the highlighted auxiliary candidates do not extend acceptance beyond the prefixes already supported by the main path and the upper branches. Once early positions have sufficient alternatives, adding more local branches provides limited improvement in accepted length. A rejection farther along the main path still terminates verification because no alternative is available at that position. Placing a short continuation at the later rejected position instead extends the illustrated accepted sequence from two to four tokens. This motivates FlexJacobi to retain the complete Jacobi path and redistribute the fixed auxiliary-node budget into short continuations along it. This design raises two challenges: identifying positions where auxiliary continuations can contribute more to accepted length and constructing effective token alternatives at those positions.

3 METHODOLOGY

To address these challenges, we propose *FlexJacobi* **asking a different question**: given limited auxiliary-node budget, which candidate continuations are most likely to translate speculative computation into practical accepted tokens? As shown in Fig 2, the core idea of FlexJacobi depends on two complementary components. **Verification-aware Topology Shaping** (VTS) determines how candidate continuations are distributed along the Jacobi trajectory, while **Context-guided Hybrid Drafting** (CHD) instantiates these continuations by combining predictive-context-guided entry selection with draft-distribution sampling. The resulting multi-candidate draft is jointly evaluated in a single target-model forward pass.

3.1 VERIFICATION-AWARE TOPOLOGY SHAPING

Recall that $Z^{(r)} = \left(z_1^{(r)}, \ldots, z_L^{(r)}\right)$ denotes the current Jacobi trajectory at iteration r with corresponding draft distributions $Q^{(r)} = \left\{q_i^{(r)}\right\}_{i=1}^{L}$. We retain all L tokens in $Z^{(r)}$ and use an auxiliary-node budget R to introduce additional candidate continuations along the trajectory, i.e., $R = B - L$, where B is a total draft-node budget. Given the current Jacobi trajectory and a limited auxiliary-node budget R, VTS determines where auxiliary continuations should be attached and how many continuations should be allocated to each position.

The key observation behind VTS is that auxiliary candidates have different potential to contribute to verification. A continuation attached at position i can only become useful when the verification process reaches that position and when additional alternatives at that position remain meaningful. Therefore, under a fixed auxiliary budget, candidate assignment should consider both the support of the preceding Jacobi trajectory and the availability of alternative predictions at the current position. Since the actual verification outcome depends on the target distributions obtained during the verification pass, it is unavailable during draft construction. VTS therefore derives an allocation priority from the current draft distributions $Q^{(r)}$.

Modeling Candidate Prioritization. Specifically, for each position i, let $m_i^0 \geq m_i^1 \geq \cdots \geq m_i^K$ denote the $K+1$ largest probabilities in the draft distribution q_i, where K is the maximum number of auxiliary continuations that can be allocated to one position. We consider two factors when assigning priority to an auxiliary continuation. First, a continuation placed at position i depends on the preceding Jacobi positions being sufficiently supported. We use $\prod_{j=1}^{i-1} m_j^0$ to measure the support of the preceding trajectory. Since m_j^0 is maximum probability, this term provides an optimistic

upper bound over the realized draft-token probabilities. Using the maximum probability avoids suppressing all later candidate positions due to a single low-probability tentative token, which may still be accepted during verification. Second, the current position should still contain sufficient probability mass for additional candidates. The value m_i^k represents the draft support associated with the k-th auxiliary allocation at position i. Larger values indicate that the current distribution still assigns non-negligible probability to additional candidate choices. Combining these two factors, VTS assigns each candidate slot (i, k) the priority score

$$s_i^k = \left(\prod_{j=1}^{i-1} m_j^0 \right) m_i^k, \qquad 1 \le i \le L, \quad 1 \le k \le K. \tag{1}$$

This score serves as shaping criterion before verification. It does not directly represent the final acceptance probability, which depends on the target distributions computed during verification.

For numerical stability, we rank candidate slots using the equivalent log form:

$$\log s_i^k = \sum_{j=1}^{i-1} \log m_j^0 + \log m_i^k.$$

The first term is the log-probability of the main-path prefix under the draft distributions. Since the target distributions are not yet available at this stage, this term serves as a draft-based proxy for the likelihood that the prefix will be accepted. The second term is the log draft probability of the k-th alternative token at position i. A position receives a higher priority when its preceding main-path tokens have high draft confidence and when the current position still has substantial draft probability mass for alternative tokens.

Allocating Candidate Continuations. Subsequently, VTS sorts all candidate slots (i, k) according to s_i^k and allocates auxiliary continuations in descending order until the auxiliary budget is exhausted. Each selected slot introduces a continuation of maximum length C, shortened only when the remaining budget is insufficient. Let K_i denote the number of continuations allocated to position i. Since $m_i^{k+1} \le m_i^k$, we have $s_i^{k+1} \le s_i^k$. Therefore, allocating additional continuations repeatedly at the same position naturally receives decreasing priority, allowing other positions along the Jacobi trajectory to compete for the limited auxiliary budget.

3.2 Context-guided Hybrid Drafting

VTS determines the positions where auxiliary continuations are introduced, but the effectiveness of each continuation depends on whether its entry token provides a suitable alternative to the main Jacobi trajectory. If the entry token is rejected during verification, subsequent tokens on the same continuation cannot contribute to the accepted sequence. To this end, we further propose Context-guided Hybrid Drafting (CHD), focusing on constructing effective continuation entries while maintaining efficient speculative drafting. Motivated by CSD (Wang et al., 2026), which observes that smooth image regions contain visual redundancy, we view previously generated visual tokens as a potential source of candidate alternatives. Different from CSD, which exploits such redundancy by relaxing verification, CHD leverages it before verification to improve candidate construction. However, historical tokens may not always match the current prediction context, and spatial proximity alone does not guarantee compatibility. CHD therefore selects candidate entries according to the consistency between the current predictive context and historical visual tokens.

Context-guided Entry Selection. For each committed image token x_j, we retain its normalized predictive representation h_j, which is derived from the final-layer states used to predict x_j. For the current Jacobi position i, we construct the query representation $\widetilde{h}_i$ from the latest available predictive state associated with this position. Additionally, let $\mathcal{I}_i$ contain all eligible committed image positions within the local neighborhood of the current position. CHD aggregates historical candidates according to:

$$w_i(v) = \sum_{j \in \mathcal{I}_i} \mathbf{1}\{x_j = v\} \exp\left(\frac{\mathrm{sim}(\widetilde{h}_i, h_j)}{\tau} \right), \tag{2}$$

where sim denotes cosine similarity and $\tau > 0$ controls the similarity weighting. The score favors tokens that have appeared under predictive contexts similar to the current one. Multiple historical

occurrences reinforce candidates that repeatedly appear in compatible contexts. We select up to K_i distinct entry tokens from retrieved candidates in descending score order. The main-path token z_i is excluded to avoid duplicating the existing Jacobi trajectory. If fewer valid historical candidates are available, the remaining entry tokens are sampled from $q_i^{(r)}$ without replacement.

Continuation Sampling from Draft Distributions. Before divergence, the candidate entry at position i follows the same prefix: $(x_{<t}, z_1^{(r)}, \ldots, z_{i-1}^{(r)})$, so the cached predictive representation $\tilde{h}_i^{(r)}$ remains valid for selecting alternative entry tokens. Once an alternative entry token is inserted, the continuation follows a different autoregressive prefix: $(x_{<t}, z_1^{(r)}, \ldots, z_{i-1}^{(r)}, y_i^{(i,k)})$. The predictive states of subsequent positions under this new prefix have not been computed. Obtaining them would require additional target-model forward passes. Therefore, CHD samples the remaining continuation tokens from the cached draft distributions: $y_j^{(i,k)} \sim q_j^{(r)}(\cdot)$. The resulting continuation combines a predictive-context-guided entry token with a draft-distribution-based continuation tail. This hybrid design applies context guidance where the required predictive representation is available, while reusing existing draft predictions after branch divergence without additional model forward passes for drafting.

3.3 Multi-Candidate Verification and Update

After VTS determines the continuation topology and CHD constructs the corresponding auxiliary continuations, the resulting multi-candidate draft is jointly evaluated by the target model(Miao et al., 2024; Cai et al., 2024). The target model computes branch-specific verification distributions according to each candidate prefix. The accepted prefix is committed following the multi-candidate SJD verification procedure. After verification, the accepted tokens and updated predictions are reused to initialize the next Jacobi iteration: $Z^{(r+1)}, \quad Q^{(r+1)}$. The predictive representations produced during the same target-model evaluation are cached for future CHD construction. FlexJacobi therefore improves multi-candidate speculative Jacobi decoding by shaping where auxiliary continuations are introduced and improving how their auxiliary candidates are drafted, while preserving the original iterative decoding framework without additional training or drafting forward passes. The pseudocode is provided in Appendix D.

4 Experiments

4.1 Experimental Setup

We evaluate FlexJacobi on Lumina-mGPT-7B (Liu et al., 2024) and Janus-Pro-7B (Chen et al., 2025). Experiments use the MS-COCO 2017 validation set (Lin et al., 2014) and PartiPrompts (Yu et al., 2022). We compare against standard AR decoding and six training-free speculative Jacobi decoding methods. Efficiency is measured by decoding latency and the number of function evaluations (NFE). We assess image quality using CLIP Score (Radford et al., 2021) on both benchmarks and FID (Heusel et al., 2017) on MS-COCO 2017. Experiments run on NVIDIA RTX 4090 GPU with FlexJacobi's total draft-node budget fixed at $B = 64$. Detailed experimental settings are provided in Appendix B.

4.2 EXPERIMENTAL RESULTS

We organize our experiments around three research questions. We assess FlexJacobi's overall performance, examine the contributions of its drafting components and evaluate its compatibility with relaxed verification.

4.2.1 Main results

RQ1 *Can FlexJacobi accelerate AR generation without sacrificing image quality?*

Quantitative results. Tables 1 and 2 summarize the main results in efficiency and image quality on Lumina-mGPT-7B (Liu et al., 2024) and Janus-Pro-7B (Chen et al., 2025), respectively. Our method achieves the lowest wall-clock latency on both Parti-Prompts (Yu et al., 2022) and MS-COCO 2017 validation set (Lin et al., 2014) while maintaining competitive generation quality.

Table 1: Decoding speed and image quality of Lumina-mGPT-7B (Liu et al., 2024) on Parti-Prompts (Yu et al., 2022) and MS-COCO 2017 (Lin et al., 2014). The champion and the runner-up are highlighted in **bold** and underline.

Method	Latency↓	NFE↓	Speedup↑		FID↓	CLIP-Score↑
			Latency	NFE		
Parti-Prompts						
Vanilla AR	68.46s	2357.00	1.00×	1.00×	—	26.86
SJD (Teng et al., 2025)	33.91s	1032.41	2.02×	2.28×	—	**26.91**
GSD (So et al., 2025)	22.22s	654.58	3.08×	3.60×	—	26.73
CSD (Wang et al., 2026)	27.22s	851.25	2.51×	2.77×	—	26.78
SCD (So et al., 2026)	21.15s	539.88	3.24×	4.37×	—	26.83
PathSpec (Lei et al., 2026)	26.88s	524.88	2.55×	4.49×	—	26.47
SJD-PAC (Kang et al., 2026)	21.21s	518.96	3.23×	4.54×	—	26.82
FlexJacobi (Ours)	**16.99s**	**458.94**	**4.03**×	**5.14**×	—	26.89
MS-COCO 2017						
Vanilla AR	68.43s	2357.00	1.00×	1.00×	**29.32**	26.34
SJD (Teng et al., 2025)	34.82s	1060.32	1.97×	2.22×	29.77	26.32
GSD (So et al., 2025)	23.08s	678.59	2.97×	3.47×	32.30	26.23
CSD (Wang et al., 2026)	28.45s	888.85	2.41×	2.65×	31.18	26.24
SCD (So et al., 2026)	22.27s	568.18	3.07×	4.15×	29.73	**26.38**
PathSpec (Lei et al., 2026)	27.70s	539.35	2.47×	4.37×	39.56	25.91
SJD-PAC (Kang et al., 2026)	22.20s	542.25	3.08×	4.35×	29.57	26.32
FlexJacobi (Ours)	**17.80s**	**480.18**	**3.84**×	**4.91**×	30.77	26.28

For Lumina-mGPT-7B, FlexJacobi reduces the decoding latency to 16.99 s on Parti-Prompts and 17.80 s on MS-COCO 2017, corresponding to 4.03× and 3.84× speedups over standard AR decoding. It also achieves the lowest NFE on both benchmarks, with 458.94 and 480.18, respectively. Compared with SJD-PAC, one of the strongest baselines, FlexJacobi consistently reduces decoding latency on both benchmarks while maintaining competitive generation quality.

For Janus-Pro-7B, FlexJacobi achieves latencies of 4.53 s and 4.85 s, giving 2.18× and 2.04× speedups over AR decoding. PathSpec attains slightly lower NFE, but requires a larger 72-node draft compared with 64 nodes in FlexJacobi, and consequently has higher wall-clock latency on both benchmarks. FlexJacobi also maintains comparable CLIP scores and achieves an FID of 26.89 on MS-COCO 2017, close to PathSpec at 26.80 and better than standard AR decoding at 27.71. Overall, these results show that FlexJacobi consistently improves decoding speed while preserving a competitive speed–quality trade-off.

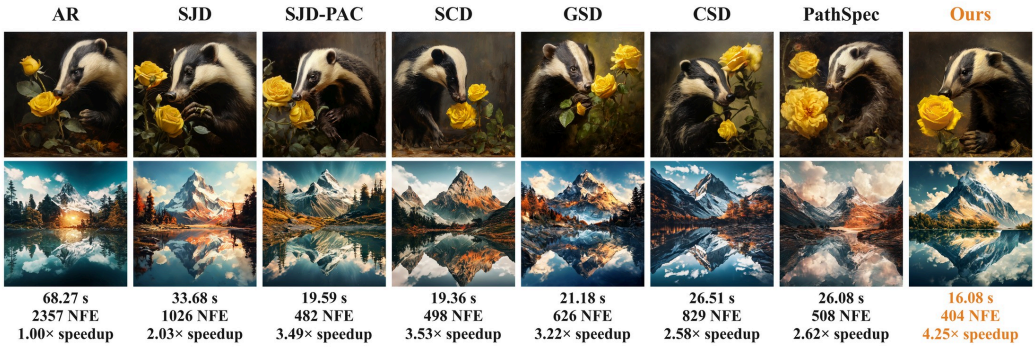


Figure 3: Qualitative comparison on PartiPrompts using Lumina-mGPT.

Qualitative Results. Figure 3 presents representative samples generated by FlexJacobi and the compared decoding methods under the same prompts. Consistent with the quantitative results,

Table 2: Decoding speed and image quality of Janus-Pro-7B (Chen et al., 2025) on Parti-Prompts (Yu et al., 2022) and MS-COCO 2017 (Lin et al., 2014). The champion and the runner-up are highlighted in **bold** and underline.

Method	Latency↓	NFE↓	Speedup↑		FID↓	CLIP-Score↑
			Latency	NFE		
Parti-Prompts						
Vanilla AR	9.88s	576.00	1.00×	1.00×	—	**26.99**
SJD (Teng et al., 2025)	6.04s	278.28	1.64×	2.07×	—	26.96
GSD (So et al., 2025)	6.25s	279.20	1.58×	2.06×	—	26.90
CSD (Wang et al., 2026)	5.01s	239.43	1.97×	2.41×	—	26.93
SCD (So et al., 2026)	6.44s	226.50	1.53×	2.54×	—	26.89
PathSpec (Lei et al., 2026)	4.97s	**162.45**	1.99×	**3.55**×	—	26.87
SJD-PAC (Kang et al., 2026)	4.94s	208.07	2.00×	2.77×	—	26.88
FlexJacobi (Ours)	**4.53s**	164.85	**2.18**×	3.49×	—	26.86
MS-COCO 2017						
Vanilla AR	9.88s	576.00	1.00×	1.00×	27.71	27.40
SJD (Teng et al., 2025)	6.38s	294.38	1.55×	1.96×	27.82	27.35
GSD (So et al., 2025)	6.61s	295.15	1.50×	1.95×	27.74	27.35
CSD (Wang et al., 2026)	5.42s	259.59	1.82×	2.22×	27.47	27.33
SCD (So et al., 2026)	6.85s	242.77	1.44×	2.37×	27.60	27.40
PathSpec (Lei et al., 2026)	5.22s	**171.81**	1.90×	**3.35**×	**26.80**	27.34
SJD-PAC (Kang et al., 2026)	5.29s	222.85	1.87×	2.59×	27.63	**27.41**
FlexJacobi (Ours)	**4.85s**	176.40	**2.04**×	3.27×	26.89	27.33

FlexJacobi preserves the main visual content and semantic alignment of the generated images while substantially reducing decoding latency. More qualitative results are provided in Appendix C.4.

> **Findings.** FlexJacobi achieves the lowest decoding latency across both models and benchmarks while maintaining competitive CLIP and FID scores.

4.2.2 Ablation Studies

RQ2 *What makes the draft construction of FlexJacobi effective?*

The overall results demonstrate the efficiency of FlexJacobi, but they do not explain where the gain comes from. To reveal the specific sources of the observed improvement, we conduct the ablation study in two stages.

Draft construction. We vary the auxiliary-node budget R without retrieval and fix the main path at 46 nodes to match SJD-PAC (Kang et al., 2026). Figure 5 shows that FlexJacobi achieves NFE speedup of $4.59\times$ with only $R = 4$, surpassing SJD-PAC with $R = 18$. NFE speedup further increases to $4.84\times$ at $R = 16$ while CLIP scores remain stable. By distributing short continuations along the main path, FlexJacobi achieves higher acceleration with fewer auxiliary nodes. These results demonstrate the effectiveness of our draft construction and its robustness across the auxiliary-node budgets. Consistent trends are observed on Janus-Pro-7B, as detailed in Appendix C.2.

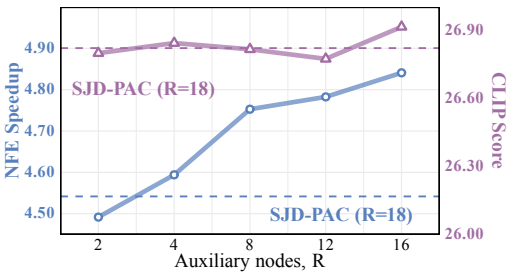


Figure 5: Effect of the auxiliary-node budget on PartiPrompts using Lumina-mGPT. Dashed lines denote the SJD-PAC configuration.



Figure 6: Efficiency comparison and retrieval ablation.

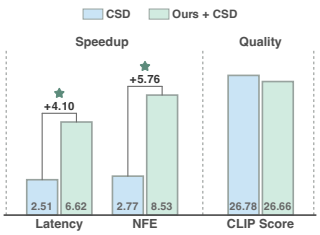


Figure 7: Combining FlexJacobi with CSD. Speedups are relative to AR.

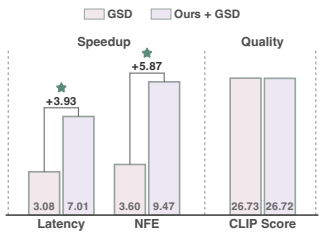


Figure 8: Combining FlexJacobi with GSD. Speedups are relative to AR.

Context-guided retrieval. We further evaluate whether context-guided token selection provides additional gains. As shown in Figure 6, sampling continuation entries from the draft distributions achieves $3.87\times$ latency speedup and $4.83\times$ NFE speedup, while adding context-guided retrieval further improves them to $4.03\times$ and $5.14\times$, respectively. This improvement shows that retrieval provides additional decoding efficiency beyond that obtained from the draft structure alone. Additional ablation studies are provided in Appendix C.1 and Appendix C.3.

> **Findings.** Constructing candidate along the main path uses the fixed verification budget more effectively than the local expansion in SJD-PAC, while context-guided retrieval provides an additional gain in MAT and decoding speed.

4.2.3 Compatibility with Relaxed Verification

RQ3 *Can FlexJacobi be combined with relaxed verification for further acceleration?*

FlexJacobi improves speculative decoding through draft construction, while methods such as CSD (Wang et al., 2026) and GSD (So et al., 2025) increase verification tolerance through relaxed acceptance rules. This raises a natural question of whether the two directions provide complementary gains. Therefore, we further examine whether FlexJacobi can be applied to existing relaxed-verification methods in a plug-and-play manner.

Figures 7 and 8 show the benefits of combining FlexJacobi with relaxed verification on Lumina-mGPT-7B (Liu et al., 2024) using Parti-Prompts (Yu et al., 2022). Both CSD and GSD achieve substantially higher speedups when equipped with FlexJacobi's draft construction. The combinations also outperform FlexJacobi alone: CSD provides an additional $1.64\times$ latency speedup, while GSD provides $1.74\times$. The corresponding NFE reductions are $1.66\times$ and $1.84\times$. These gains show that the better draft construction remains effective under relaxed acceptance rules. For generation quality, only a minor decrease in CLIP Score is observed for the combined methods. Compared to CSD and GSD alone, combined methods reduce the CLIP score by only 0.12 and 0.01, respectively. These results demonstrate the compatibility of FlexJacobi with relaxed verification methods for further acceleration.

> **Findings.** FlexJacobi and relaxed verification are complementary: replacing the verification rule further reduces latency and NFE while causing only small changes in CLIP score.

5 Conclusion

In this paper, we introduced FlexJacobi, a training-free framework for improving auxiliary-candidate efficiency in multi-candidate SJD. Verification-aware Topology Shaping allocates short continuations along the complete Jacobi path under a fixed budget, while Context-guided Hybrid Drafting combines predictive-context-guided entry selection with cached draft-distribution sampling. Experiments demonstrate that FlexJacobi achieves lower decoding latency and competitive quality with far fewer auxiliary nodes, and remains compatible with relaxed verification, showing that efficient speculation hinges on candidate allocation and construction rather than candidate count alone.

AI USE STATEMENT

We used generative AI tools to aid and polish writing. We have reviewed all AI-assisted work. AI-assisted writing was manually checked and revised by the authors to ensure that the final text accurately reflects the intended technical content and claims. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *International Conference on Machine Learning*, pp. 5209–5235, 2024.

Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning*, pp. 1691–1703, 2020.

Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model. In *International conference on machine learning*, pp. 864–872, 2018.

Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. Janus-pro: Unified multimodal understanding and generation with data and model scaling. *arXiv preprint arXiv:2501.17811*, 2025.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable and robust speculative decoding. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems*, 2024.

Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *Computer Vision and Pattern Recognition*, pp. 12873–12883, 2021.

Oran Gafni, Adam Polyak, Oron Ashual, Shelly Sheynin, Devi Parikh, and Yaniv Taigman. Make-a-scene: Scene-based text-to-image generation with human priors. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XV*, pp. 89–106, 2022.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pp. 6626–6637, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jialiang Kang, Han Shu, Wenshuo Li, Yingjie Zhai, and Xinghao Chen. Sjd-pac: Accelerating speculative jacobi decoding via proactive drafting and adaptive continuation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Computer Vision and Pattern Recognition*, pp. 11513–11522, 2022.

Haodong Lei, Hongsong Wang, Bingxuan Dai, and Pan Zhou. Pathrelax: Parallel-path relaxed speculative jacobi decoding for accelerating auto-regressive text-to-image generation. *arXiv preprint arXiv:2606.10492*, 2026.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286, 2023.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: faster inference of language models with dynamic draft trees. In *Empirical Methods in Natural Language Processing*, pp. 7421–7432, 2024.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *Computer Vision - ECCV 2014 - 13th European Conference*, pp. 740–755, 2014.

Dongyang Liu, Shitian Zhao, Le Zhuo, Weifeng Lin, Yi Xin, Xinyue Li, Qi Qin, Yu Qiao, Hongsheng Li, and Peng Gao. Lumina-mgpt: Illuminate flexible photorealistic text-to-image generation with multimodal generative pretraining. *arXiv preprint arXiv:2408.02657*, 2024.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 932–949, 2024.

Sihwan Park, Doohyuk Jang, Sungyub Kim, Souvik Kundu, and Eunho Yang. Lantern++: Enhancing relaxed speculative decoding with static tree drafting for visual auto-regressive models. *arXiv preprint arXiv:2502.06352*, 2025.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International conference on machine learning*, pp. 4055–4064, 2018.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763, 2021.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pp. 8821–8831, 2021.

Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*, 2019.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *International Conference on Learning Representations*, 2017.

Junhyuk So, Juncheol Shin, Hyunho Kook, and Eunhyeok Park. Grouped speculative decoding for autoregressive image generation. In *International Conference on Computer Vision*, pp. 15375–15384, 2025.

Junhyuk So, Hyunho Kook, Chaeyeon Jang, and Eunhyeok Park. Speculative coupled decoding for training-free lossless acceleration of autoregressive visual generation. In *Proceedings of the International Conference on Machine Learning*, 2026.

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

Yao Teng, Han Shi, Xian Liu, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. Accelerating auto-regressive text-to-image generation with training-free speculative jacobi decoding. In *International Conference on Learning Representations*, 2025.

Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*, pp. 4790–4798, 2016.

Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pp. 1747–1756, 2016.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pp. 6306–6315, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pp. 5998–6008, 2017.

Mingcheng Wang, Junbo Qiao, Yunchen Li, Lingfu Jiang, Wei Li, Jie Hu, Jiao Xie, Zhou Yu, Xinghao Chen, Guixu Zhang, et al. Csd: Content-aware speculative decoding for efficient image generation. *arXiv preprint arXiv:2606.27829*, 2026.

Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems*, 2024.

Baoquan Zhang, Bingqi Shan, Shihao Fang, Kenghong Lin, Xutao Li, and Yunming Ye. Sjd-sv: Speculative jacobi decoding with semantics verification for autoregressive image generation. *arXiv preprint arXiv:2609.13245*, 2026.

CONTENTS

A. Related work **14**

A.1. Autoregressive Image Generation 14

A.2. Speculative Jacobi Decoding for Image Generation 14

B. Additional Experimental Details **14**

B.1. Models and Benchmarks 14

B.2. Baselines 15

B.3. Evaluation Metrics 15

B.4. Implementation details 16

C. Additional Experimental Results **16**

C.1. Effect of hyperparameters 16

C.2. Additional Ablations on Draft Construction 17

C.3. Additional Ablations on retrieval 18

C.4. Additional Qualitative Results 18

D. Algorithm **19**

A Related work

A.1 Autoregressive Image Generation

Early autoregressive image models factorized the image distribution over pixels and generated them sequentially (Van Den Oord et al., 2016; van den Oord et al., 2016; Salimans et al., 2017; Chen et al., 2018; Parmar et al., 2018; Chen et al., 2020). Subsequent approaches adopted a two-stage formulation, using a learned tokenizer (van den Oord et al., 2017; Razavi et al., 2019; Yu et al., 2024) to compress images into discrete visual codes and a Transformer (Vaswani et al., 2017) to model the resulting sequence through next-token prediction (Ramesh et al., 2021; Sun et al., 2024; Esser et al., 2021; Lee et al., 2022). More recent unified models, including Chameleon (Chameleon Team, 2024), Emu3 (Wang et al., 2024), Lumina-mGPT (Liu et al., 2024), and Janus-Pro (Chen et al., 2025), bring visual understanding and image generation into a common autoregressive framework. Despite advances in visual representation and multimodal training, conventional inference in these models still generates image tokens one at a time. The long visual sequences required for image synthesis therefore entail many successive model evaluations, making decoding latency a persistent bottleneck.

A.2 Speculative Jacobi Decoding for Image Generation

Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) accelerates autoregressive generation through lightweight drafting and parallel target-model verification. A draft model proposes multiple tokens that the target model evaluates in a single forward pass. An acceptance-and-resampling procedure preserves the target distribution while allowing multiple tokens to be committed per iteration.

Speculative Jacobi Decoding (SJD) (Teng et al., 2025) adapts speculative decoding to autoregressive image generation by reusing previous target-model predictions as draft distributions. It refines a window of tentative tokens in parallel without a separate draft model. Within this framework, SCD (So et al., 2026) reduces unnecessary token changes across iterations by coupling successive sampling steps. This increases token reuse while preserving the required sampling marginals.

While SCD improves consistency across drafting iterations, GSD (So et al., 2025) and CSD (Wang et al., 2026) exploit visual redundancy to relax verification criteria at the verification stage, allowing more draft tokens to be accepted. GSD forms context-dependent token groups and evaluates their aggregated probability mass rather than relying solely on individual token probabilities. CSD uses target-distribution entropy to adapt acceptance relaxation to image content. A total-variation filter restricts relaxation to positions where draft and target distributions are sufficiently aligned. At the same time, SJD-SV (Zhang et al., 2026) further accounts for semantic relationships between visual tokens by verifying semantic-aware subsequences instead of individual tokens. These methods improve verification by changing the acceptance criterion or its granularity.

Beyond modifying sampling and relaxing verification, multi-candidate methods change the organization of the draft itself (Cai et al., 2024; Li et al., 2024; Chen et al., 2024; Park et al., 2025). SJD-PAC (Kang et al., 2026) constructs a shallow local K-ary tree after rejection and extends one path into a long chain; its adaptive continuation mechanism also retains useful unresolved states for subsequent iterations. PathRelax (Lei et al., 2026) explores parallel Jacobi paths and shares probability mass among sibling candidates for relaxed verification. However, the local expansion pattern in SJD-PAC and the per-level capacity constraints in PathRelax prescribe where candidate coverage is available.

B Additional Experimental Details

B.1 Models and Benchmarks

We conduct experiments on two autoregressive text-to-image models, Lumina-mGPT (Liu et al., 2024) and Janus-Pro (Chen et al., 2025), generating images at resolutions of 768×768 and 384×384 pixels, respectively. We evaluate both models on the MS-COCO 2017 validation set (Lin et al., 2014) and the standard PartiPrompts benchmark (Yu et al., 2022). MS-COCO provides natural images paired with human-annotated captions, covering diverse objects, scenes, and compositional

relationships. PartiPrompts complements this evaluation with prompts spanning semantic categories, attributes, spatial relationships, and compositional concepts. For each model, all compared decoding methods use the same pretrained checkpoint, image resolution, and sampling configuration to ensure a fair comparison.

B.2 Baselines

We compare with six training-free speculative Jacobi decoding methods: SJD (Teng et al., 2025), GSD (So et al., 2025), CSD (Wang et al., 2026), SCD (So et al., 2026), PathSpec (Lei et al., 2026) and SJD-PAC (Kang et al., 2026). These baselines cover both chain-based and tree-structured drafting, as well as standard and relaxed acceptance rules. We use the officially released implementations and follow the best-performing configurations reported in the corresponding papers. Here, we provide detailed descriptions of the baselines evaluated in our experiments.

- **AR (Autoregressive Decoding)** samples each token directly from the target model conditioned on the generated prefix. It produces one token per forward pass without speculative drafting. This is the reference decoding procedure for measuring latency and NFE speedups.
- **SJD (Speculative Jacobi Decoding)** combines parallel Jacobi refinement with speculative verification. It reuses predictions from previous target-model evaluations as draft distributions without requiring a separate draft model. A single forward pass evaluates the current draft window. Verification commits the accepted prefix while the remaining positions are updated for subsequent iterations.
- **GSD (Grouped Speculative Decoding)** relaxes token-level verification by comparing the aggregated probability mass of token groups. Groups are formed dynamically from the current token probabilities. Embedding-distance and probability-difference thresholds exclude unsuitable group members. This group-based acceptance rule increases acceptance of alternative visual tokens within a training-free Jacobi decoding framework.
- **CSD (Content-aware Speculative Decoding)** adapts token acceptance to image content through entropy-based probability relaxation. It applies stronger relaxation in smooth regions with higher target-distribution entropy to reduce redundant computation. A distribution alignment filter enables relaxation only when the total variation distance between draft and target distributions falls below a threshold. On rejection, it retains residual resampling to minimize total-variation deviation from the target distribution under the relaxed acceptance rule.
- **SCD (Speculative Coupled Decoding)** reduces unnecessary token changes across successive Jacobi iterations through coupled sampling. The method considers maximal coupling and Gumbel coupling to increase token reuse without changing the required sampling marginals. This improves the consistency of consecutive draft contexts. Standard speculative verification is retained to preserve the target distribution.
- **PathSpec** extends single-chain SJD to multiple candidate paths sharing a committed prefix. Its PathExplore component evaluates these paths using tree attention and selects the longest accepted prefix. The framework also introduces PathRelax for cross-path relaxed verification. This component adjusts acceptance through probability sharing among sibling candidates with the same parent prefix.
- **SJD-PAC** addresses frequent early rejections in SJD through proactive drafting and adaptive continuation. Proactive drafting constructs a shallow local candidate tree after rejection and extends one path into a long chain. Adaptive continuation applies acceptance checks to the remaining uncommitted positions to retain reusable tokens rather than independently resampling all of them. These updates prepare the next draft without committing additional output in the current iteration.

B.3 Evaluation Metrics

We evaluate FlexJacobi in terms of decoding efficiency and generation quality. The metrics used in our experiments are described below.

Decoding efficiency. We report decoding latency and the number of function evaluations. MAT is additionally used to evaluate auxiliary-candidate efficiency in the ablation study.

- **Latency** measures the average autoregressive decoding time per image in seconds. It excludes the subsequent conversion of visual tokens into pixels. This metric evaluates the actual decoding time required by each method. Lower latency indicates faster decoding.
- **NFE (Number of Function Evaluations)** measures the average number of sequential target-model forward steps required to generate an image. Unlike latency, NFE measures the number of evaluations rather than their execution time. A lower NFE indicates that generation requires fewer sequential decoding steps.
- **MAT (Mean Accepted Tokens)** measures the average number of tokens committed per verification step. It evaluates the extent to which draft candidates contribute to the generated sequence. A higher MAT indicates that more tokens are committed in each verification step.

For latency and NFE, speedup is calculated as the ratio of the standard AR baseline value to that of the evaluated method.

Generation quality. We assess text–image alignment using CLIP Score and compare image distributions using FID.

- **CLIP Score** measures the semantic alignment between generated images and their corresponding text prompts. We use the ViT-L/14 CLIP model (Radford et al., 2021) to extract image and text embeddings. Their cosine similarity provides the alignment score. We report CLIP Score on both MS-COCO 2017 and PartiPrompts. Higher scores indicate better alignment between the generated images and the prompts.
- **FID (Fréchet Inception Distance)** (Heusel et al., 2017) measures the discrepancy between generated and real image distributions in the Inception feature space (Szegedy et al., 2016). We evaluate FID on MS-COCO 2017 using real images from the validation set as the reference. Lower FID indicates that the generated image distribution is closer to the reference distribution.

Since PartiPrompts does not provide ground-truth images, we report only CLIP Score for this benchmark.

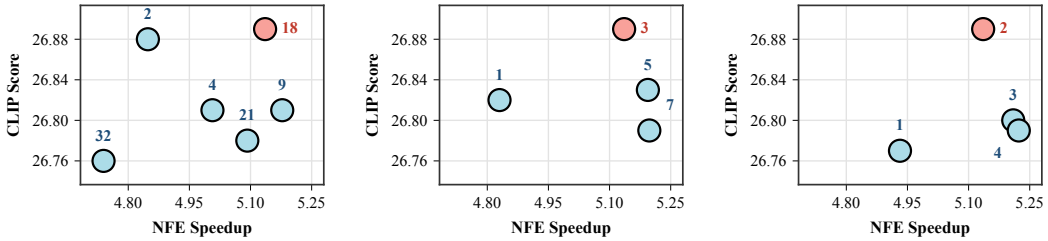
B.4 Implementation details

All experiments are conducted on NVIDIA RTX 4090 GPU. For Lumina-mGPT-7B, we use top-k sampling with $k = 2000$, a temperature of 1.0, and a Classifier-Free Guidance (CFG) (Ho & Salimans, 2022; Gafni et al., 2022) scale of 3. For Janus-Pro-7B, we follow its default generation configuration with a CFG scale of 5. We set the same total draft-node budget to $B = 64$ for these two models. Considering the different generation regimes of the two models, we use separate tree configurations for long- and short-context generation. For Lumina-mGPT, which represents the long-context setting with 2,357 NFEs under standard AR decoding, For Lumina-mGPT, we use a main candidate path of length $L = 46$ and $R = 18$ auxiliary candidate nodes, with $C = 2$ and $K = 3$. For Janus-Pro, we set $L = 24$ and $R = 40$, with $C = 4$ and $K = 7$.

C Additional Experimental Results

C.1 Effect of hyperparameters

Auxiliary-node budget R. We vary R under a fixed total budget of $B = 64$, with the main-path length set to $L = B - R$. Figure 9a shows that NFE speedup initially increases with R but declines at larger budgets. A small auxiliary budget provides few alternatives when a main-path token is rejected. Increasing R provides more opportunities to continue verification along auxiliary paths. However, further increasing R shortens the main path while offering limited additional gains in accepted length. This trend suggests a trade-off between main-path length and the number of auxiliary candidates.



(a) The number of auxiliary nodes R

(b) The maximum number of continuations per position K

(c) The continuation length C

Figure 9: Hyperparameter ablation of **FlexJacobi** on Lumina-mGPT: (a) the number of auxiliary nodes R, (b) the maximum number of continuations per position K, and (c) the continuation length C. The red one is the default setting.

Maximum continuations per position K. With the auxiliary budget R and continuation length C fixed, increasing K allows more auxiliary continuations to be allocated to the same main-path position, providing additional alternatives when verification encounters a rejection. The improvement gradually saturates as K increases. Since the total number of continuations is bounded by the fixed auxiliary budget, a larger K mainly permits more candidates to concentrate at individual positions rather than increasing the overall candidate capacity. As more alternatives are already available at a position, additional continuations provide diminishing marginal benefits in extending the accepted sequence. This trend is shown in Figure 9b.

Continuation length C. With the auxiliary budget R fixed, increasing C allows each auxiliary continuation to support a longer accepted sequence once its entry candidate is accepted. The improvement, however, gradually saturates as C increases. A longer continuation consumes more auxiliary nodes, reducing the number of continuations that can be distributed along the main path. Moreover, deeper continuation tokens contribute only when all preceding tokens on the same branch are accepted, resulting in progressively smaller marginal benefits. This trade-off between continuation depth and the distribution of auxiliary continuations explains the observed saturation at larger C. This behavior is illustrated in Figure 9c.

The default configuration $(R, K, C) = (18, 3, 2)$ achieves NFE speedup close to the highest values in all three sweeps. It also obtains the highest observed CLIP Score. We retain this configuration for its balance between decoding efficiency and generation quality.

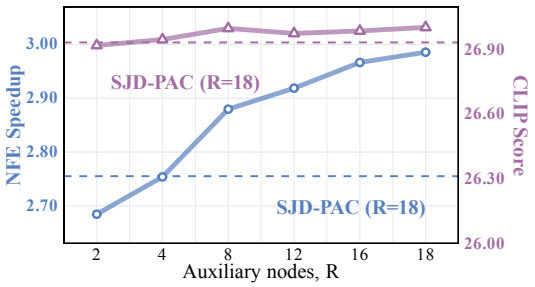


Figure 10: Effect of the auxiliary-node budget with a fixed main path for Janus-Pro 7B. Dashed lines denote the SJD-PAC configuration.

C.2 Additional Ablations on Draft Construction

We further evaluate the auxiliary-node budget R on Janus-Pro-7B while keeping the main path fixed. Figure 10 shows the same trend observed on Lumina-mGPT: FlexJacobi matches SJD-PAC with substantially fewer auxiliary nodes and surpasses it as the budget increases, while the CLIP Score

Table 3: Ablation of retrieval and candidate-sampling strategies on Lumina-mGPT-7B using Parti-Prompts.

Method	**Latency↓**	**NFE↓**	**MAT↑**	**CLIP-Score↑**
ours+Retrieval	16.99	458.94	5.02	26.89
ours+Uniform local-history sampling	16.57	442.53	5.20	26.44
ours+Q-sampling	17.71	487.95	4.72	26.83

remains stable. This consistent behavior across the two models further validates the effectiveness of our draft structure in utilizing auxiliary candidates more efficiently.

C.3 Additional Ablations on Retrieval

We further compare three candidate-construction strategies under the same 64-node budget and identical topology, verification procedure, and main-path update rule. *Retrieval* uses hidden-state similarity to rank historical candidates, *Uniform local-history sampling* uniformly samples distinct historical image tokens from the same causally valid local-history region without similarity ranking, and *Q-sampling* samples side-branch heads directly from the draft distribution without retrieval. The key comparison is between *ours+Retrieval* and *ours+Uniform local-history sampling*: both use the same causally valid local-history region, while the former ranks historical image tokens by hidden-state similarity and the latter samples them uniformly without similarity ranking. Although uniform local-history sampling provides a modest speedup, its CLIP score drops substantially to 26.44. In contrast, the retrieval variant achieves a CLIP score essentially on par with Q-sampling. Table 3 shows that hidden-state retrieval is important for selecting semantically aligned candidates and preserving the image quality.

C.4 Additional Qualitative Results

We provide additional qualitative comparisons to complement the results in the main text. Figures 11–14 show more generated samples under the same prompts across different decoding methods.

D ALGORITHM

Algorithm 1 FlexJacobi

Input: Root a, updated main candidate path z, draft distributions q, queries $\tilde{h}$, retrieval bank $\mathcal{B}$
Input: Auxiliary-node budget R, maximum continuations per position K, continuation length C

$L \leftarrow |z|$; $\mathcal{T} \leftarrow (a, z_1, \ldots, z_L)$
$K_i \leftarrow 0, i = 1, \ldots, L$ ▷ Auxiliary continuations allocated at position i
$(m_i^0, \ldots, m_i^K) \leftarrow \textsc{TopProb}(q_i, K+1), \forall i$ ▷ Largest probabilities, descending
$\log s_i^k \leftarrow \sum_{j=1}^{i-1} \log m_j^0 + \log m_i^k, 1 \le i \le L, 1 \le k \le K$
$\mathcal{A} \leftarrow \textsc{Sort}(\log s_i^k)$ ▷ Pairs (i, k) by descending finite score
for $(i, k) \in \mathcal{A}$ **do**
  if $R = 0$ **then**
    break
  end if
  $\mathcal{T} \leftarrow \textsc{AttachChain}(\mathcal{T}, i, \min(C, R))$ ▷ Branch from the parent of z_i
  $K_i \leftarrow K_i + 1$; $R \leftarrow R - \min(C, R)$
end for
for $i = 1, \ldots, L$ with $K_i > 0$ **do**
  $w_i \leftarrow \textsc{Retrieval}(i, \tilde{h}_i, \mathcal{B})$ ▷ Local retrieval and token voting
  $Y_i \leftarrow \textsc{SelectCandidates}(w_i, q_i, z_i, K_i)$ ▷ Rank by w_i; sample to fill K_i
    ▷ Exclude z_i, duplicates
  for $k = 1, \ldots, K_i$ **do**
    $y_i^{(i,k)} \leftarrow Y_i[k]$ ▷ First token of this continuation
    $y_j^{(i,k)} \sim q_j$ for remaining nodes on this continuation
  end for
end for
return $\mathcal{T}$

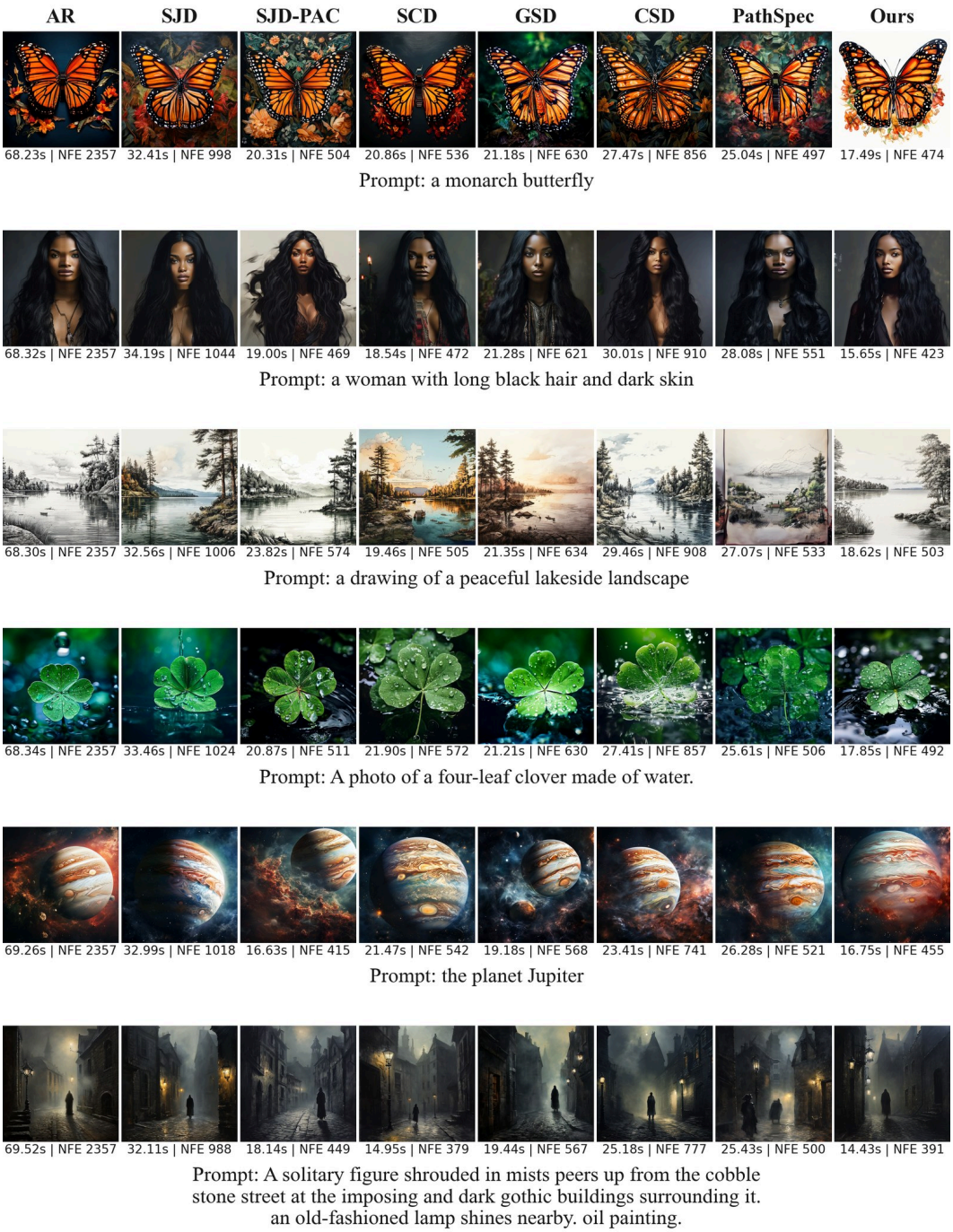


Figure 11: More qualitative comparison on PartiPrompts using Lumina-mGPT



Figure 12: More qualitative comparison on MS-COCO2017 using Lumina-mGPT

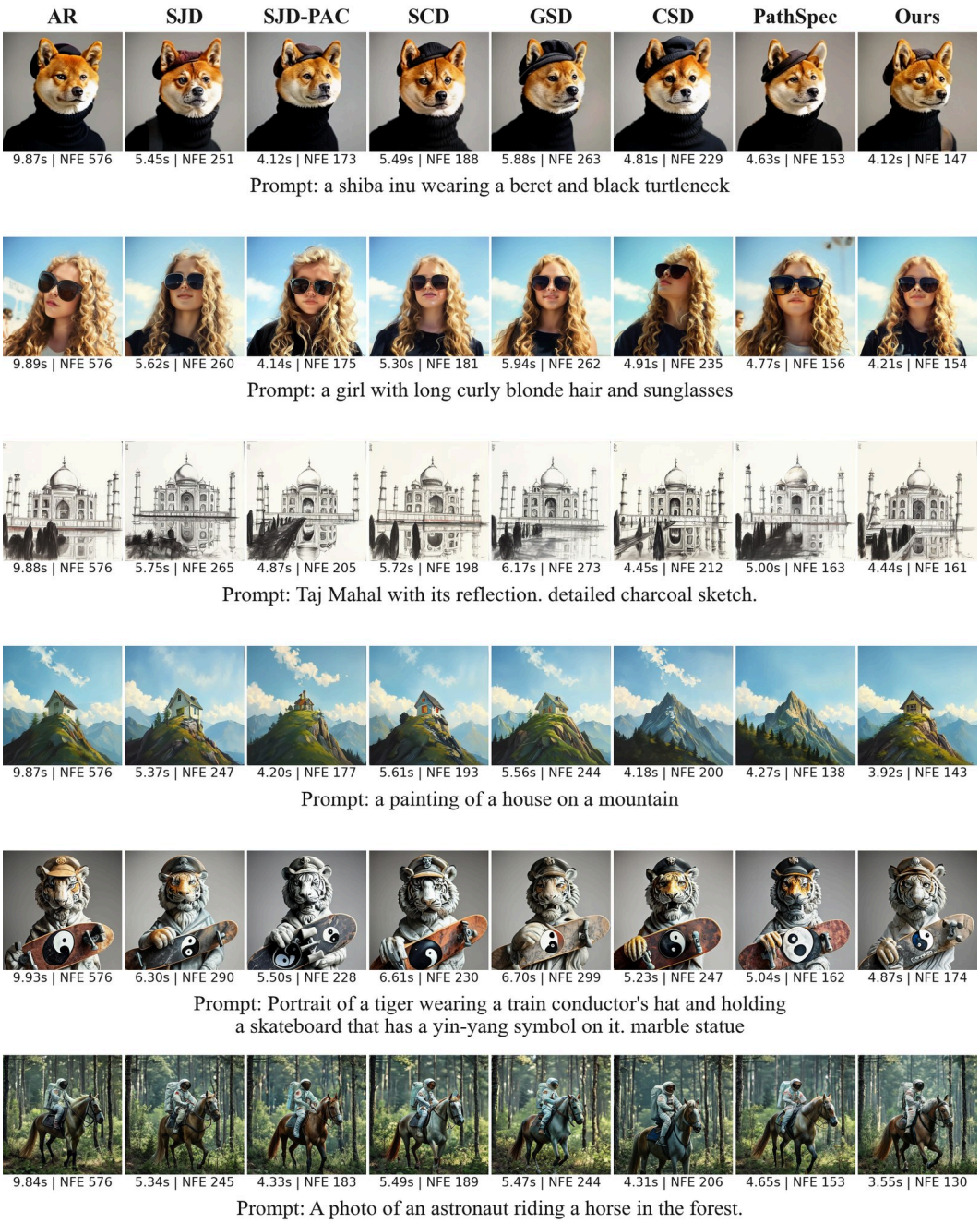


Figure 13: More qualitative comparison on PartiPrompts using Janus-Pro 7B



Figure 14: More qualitative comparison on MS-COCO2017 using Janus-Pro 7B