SSD: SPATIALLY SPECULATIVE DECODING ACCELERATES AUTOREGRESSIVE IMAGE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive image models treat images as 1D token sequences, inheriting the next-token factorization of language models. This flattening discards a useful property of images: visual tokens are organized and correlated in two-dimensional space. We introduce Spatially Speculative Decoding (SSD), an inference-time decoding framework that exploits this spatial structure for accelerated generation. Rather than speculating only along the flattened sequence, SSD predicts both the adjacent horizontal token and the token directly below it, allowing multiple spatial directions to advance in parallel. This reduces the number of backbone forward evaluations and alleviates the memory bottleneck of autoregressive decoding. SSD accelerates image generation by up to 11.03× in wall-clock time while maintaining generation quality on DPG-Bench and GenEval. These results show that spatial structure provides a simple and effective source of parallelism for faster autoregressive image generation.

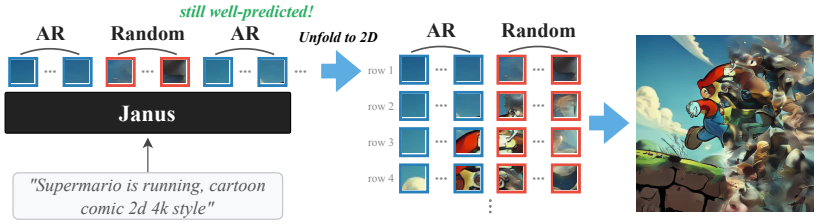
1 INTRODUCTION

The appeal of autoregressive visual generation lies in its unification: a single "next-token" objective modeling both human text and the visual world (Wu et al., 2025; Chen et al., 2025; Sun et al., 2024b; Xie et al., 2025), unlocking the potential for multimodal chain-of-thought reasoning and beyond (Zhang et al., 2024). However, this modality-agnostic architecture introduces a geometric compromise. To fit the linear processing stream of language models, the two-dimensional fabric of images is flattened into a one-dimensional sequence (Team, 2024; Hurst et al., 2024). While this linguistic formulation allows vision and text to share a statistical engine, it discards the spatial priors of the physical world (Esser et al., 2021; Ramesh et al., 2021; Yu et al., 2022; Sun et al., 2024a; Wang et al., 2024). Synthesizing a single image requires generating thousands of discrete visual tokens, making strict next-token prediction inherently slow. The length of these flattened sequences creates a severe inference bottleneck for unified autoregressive models.

A large body of work attempts to accelerate this bottleneck by importing natural language processing techniques. Grafting speculative decoding (Leviathan et al., 2023; Chen et al., 2023; Li et al., 2024a; Cai et al., 2024; Li et al., 2024b) and Jacobi-based iteration (Fu et al., 2024; Santilli et al., 2023) onto visual models yields modest 1.8× to 3.7× speedups (Jang et al., 2025; So et al., 2025; Teng et al., 2025; Peruzzo et al., 2026; Li et al., 2026b). However, these adaptations remain bound to the 1D sequential assumption. Unlike text, visual patches exhibit high localized entropy without rigid grammar; anticipating multiple tokens along a flattened 1D horizon thus yields low draft acceptance rates (Jang et al., 2025; So et al., 2025). Other methods introduce spatial parallelism, but often rely on restrictive independence assumptions (He et al., 2025a; Wang et al., 2025) or require training new architectures from scratch (He et al., 2025b; Zhang et al., 2025; Li et al., 2026a). Recent approaches relax exact token-level equivalence during speculative decoding (He et al., 2025a; Jang et al., 2025; Li et al., 2026b), while maintaining the generation quality.

In this paper, we introduce Spatially Speculative Decoding (SSD), a framework that aligns the mechanics of autoregressive generation with the intrinsic structure of vision domain. Our key insight is that spatial correlations are inherently two-dimensional: within a raster-scan sequence, the token directly below a position shares a predictive dependency comparable to the immediate next token (Figure 1). Rather than guessing a fragile 1D future, our approach simultaneously anticipates visual space across horizontal and vertical axes (Figure 2). By respecting this 2D locality, the model

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.



Accept Rate (%)		
Off.	Horiz.	Vert.
+1	43.27	30.32
+2	29.69	27.03
+3	25.47	25.71

Figure 1: **The Two-Dimensional Nature of Predictive Dependency.** To demonstrate that spatial correlations are inherently 2D, we corrupt the sequential context during Janus-Pro-7B generation by replacing the second half of each row with random tokens (red outlines). Despite this severe disruption to the 1D sequence, visual coherence is preserved wherever the token directly *above* was accurately generated (blue outlines). This confirms that vertical prediction relies fundamentally on spatial adjacency rather than position in the flattened raster-scan order. **(Right)** Acceptance rates of horizontal vs. vertical drafting heads at matching spatial offsets, confirming that predictability is governed by 2D spatial locality.

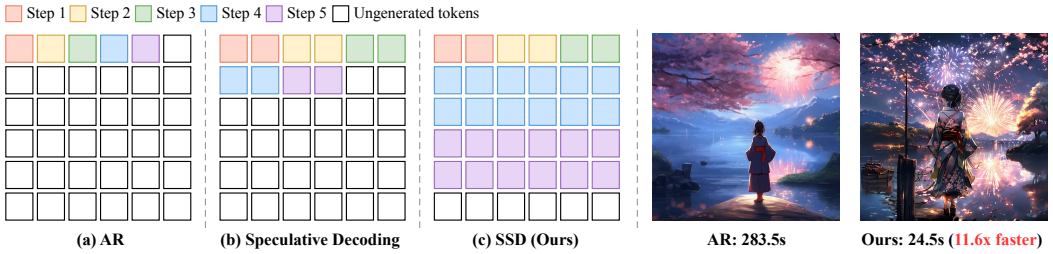


Figure 2: **Accelerating Autoregressive Vision via 2D Spatial Anticipation.** (a) Standard AR flattens the visual world into a 1D sequence and predicts one token per backbone forward evaluation. (b) Speculative Decoding accelerates generation locally but remains constrained by this linear raster-scan geometry. (c) Our SSD aligns the predictive objective with the intrinsic geometry of images and drafts entire spatial blocks in parallel. This reduces the number of backbone forward evaluations and alleviates the memory bottleneck of autoregressive decoding. Right: Applied to Emu3 (8B), this geometric shift yields a 11.03× speedup while preserving high-resolution visual fidelity.

drafts entire rows at once. This geometric shift reduces the number of forward evaluations (NFE) complexity, which is the key bottleneck for inference speed.

We further reduce verification cost by following the non-exact acceptance schemes used in visual speculative decoding He et al. (2025a); Jang et al. (2025); Li et al. (2026b). SSD drafts many spatially correlated tokens at once, so truncating the draft at the first mismatch can discard many useful predictions. We instead use the verification pass to correct the draft: mismatched tokens are replaced with predictions from the backbone, and the resulting block can be verified again. Repeating this correction gives a simple knob between speed and adherence to the original model distribution. Empirically, two verification passes already produce images visually indistinguishable from standard autoregressive decoding (Figure 8).

A key advantage of our framework is that it factors out inference acceleration from backbone pretraining. Because SSD requires no modifications to the pretrained backbone, it serves as a plug-and-play module that accelerates visual generation when needed and leaves the original model untouched when removed. Moreover, the modularity of our approach allows it to generalize to any unified autoregressive model that produces discrete visual tokens. Because these drafting heads are lightweight, the FLOPs overhead is minimal (Figure 4), enabling massive computational savings by successfully overcoming the memory wall.

Figure 3 illustrates this speedup with a single generation example on Janus-Pro. At the moment SSD completes all 576 image tokens, autoregressive decoding and existing acceleration methods have produced only partial images. Across the full evaluation, SSD accelerates wall-clock generation by

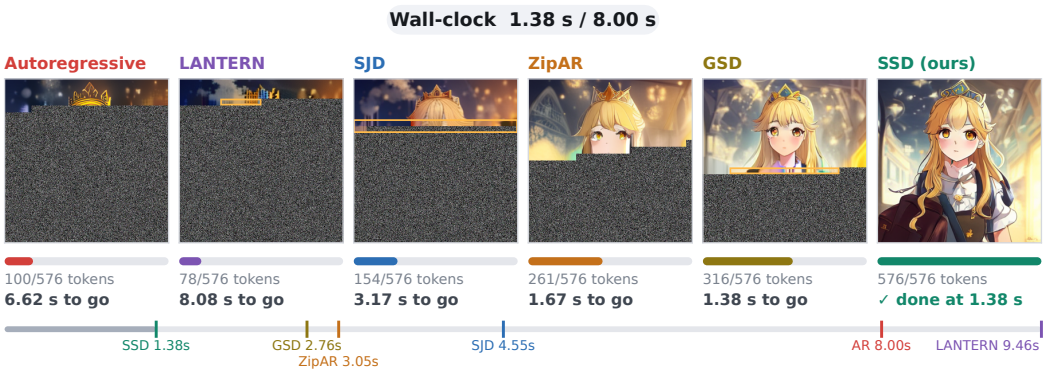


Figure 3: **Generation Progress at a Matched Wall-Clock Budget.** The snapshots show Janus-Pro-7B generation for the same prompt at the instant SSD finishes (1.38 s): SSD has completed all 576 image tokens, whereas autoregressive decoding and existing state-of-the-art acceleration methods remain largely incomplete. The bottom axis marks the total latency at which each method finishes. The snapshots and completion markers provide an intuitive illustration of the substantial speedup achieved by SSD over all baselines.

7.35× on Lumina-mGPT (Liu et al., 2024b), 5.70× on Janus-Pro (Chen et al., 2025), and 11.03× on Emu3 (Wang et al., 2024). Our results show that 1D sequential decoding is a costly artifact of the language-model formulation rather than a necessity for image generation. By exploiting the geometry of images, SSD removes much of this unnecessary sequential computation.

2 Related Work

Autoregressive Image Generation. Discrete visual tokenization approaches, such as VQ-VAE (Van Den Oord et al., 2017), VQGAN (Esser et al., 2021), and MAGVIT-v2 (Yu et al., 2024), allow images to be represented as discrete latent features. This discrete representation enables standard transformer architectures to synthesize images via autoregressive next-token prediction (Ramesh et al., 2021; Yu et al., 2022; Sun et al., 2024a). Furthermore, this shared discrete formulation has catalyzed the development of unified vision-language models (Wu et al., 2025; Chen et al., 2025; Sun et al., 2024b; Xie et al., 2025; Team, 2024; Wang et al., 2024) capable of performing both understanding and generation within a single backbone.

Multi-Token Prediction and Speculative Decoding. Multi-token prediction (MTP) (Gloeckle et al., 2024) has emerged as an effective strategy to bypass this memory wall. By utilizing auxiliary heads to anticipate multiple future tokens during a single forward pass, MTP significantly reduces the number of required backbone forward passes. To ensure the output quality of these MTP drafts matches the target distribution, speculative decoding (Leviathan et al., 2023; Chen et al., 2023) verifies candidate tokens in a single parallel pass (Cai et al., 2024; Li et al., 2024a;b; 2025; Miao et al., 2024). Because this verification is computed in parallel across all draft tokens, it successfully overcomes the memory wall by requiring only a single forward pass for verification. Similarly, Jacobi-based methods (Fu et al., 2024; Santilli et al., 2023; Kou et al., 2024) build upon this parallel verification principle by framing decoding as fixed-point convergence achieved via iterative forward passes, thereby eliminating the need for a separate draft model. While these techniques yield speedups in text generation, their application to visual domains is limited by the one-dimensional sequential setting, as they only anticipate subsequent tokens along a flattened 1D raster-scan order.

Accelerating Image Autoregressive Generation. Consequently, directly applying these 1D-centric acceleration techniques to visual autoregressive models (Jang et al., 2025; So et al., 2025; Teng et al., 2025; Peruzzo et al., 2026; Li et al., 2026b) yields marginal improvements, typically plateauing at 1.8× to 3.7× speedups. Recent methods explore spatial parallelism under two distinct settings. Existing plug-in acceleration methods typically keep the pretrained backbone frozen and introduce only lightweight auxiliary prediction heads, but achieve limited speedups (Jang et al., 2025; So et al., 2025; Teng et al., 2025; Peruzzo et al., 2026; Li et al., 2026b). Other methods (He et al., 2025b;

Wang et al., 2025; Zhang et al., 2025; Li et al., 2026a; Zhou et al., 2026) pursue stronger parallelism by modifying and retraining the backbone. While effective, backbone retraining is costly for large foundation models and may alter their pretrained capabilities. PJD (Liao et al., 2026) instead extends Jacobi decoding to two-dimensional row-wise refinement without learning spatial drafts. In contrast, our SSD preserves the pretrained backbone and learns only lightweight spatial drafting heads, while exploiting 2D spatial structure to achieve substantially larger acceleration.

3 Method: Spatially Speculative Decoding

3.1 Autoregressive Visual Generation in Unified Vision Language Models

Modern autoregressive image models (Ramesh et al., 2021; Yu et al., 2022; Sun et al., 2024a) typically consist of two components: a vector quantizer (VQ) (Van Den Oord et al., 2017; Esser et al., 2021) that discretizes images into tokens, and an autoregressive transformer (Vaswani et al., 2017) that models the resulting sequence. The VQ encoder $\mathcal{E}$ maps a continuous image $\mathbf{X} \in \mathbb{R}^{H\times W\times 3}$ to a grid of latent features, each quantized to its nearest entry in a learned codebook $\mathcal{V}$, yielding a discrete token grid $\mathbf{T} \in \mathcal{V}^{n\times n}$. A decoder $\mathcal{D}$ reconstructs the image from $\mathbf{T}$.

Given this discrete representation, image generation reduces to sequential token prediction. The grid $\mathbf{T}$ is flattened into a one-dimensional sequence $(t_1, t_2, \ldots, t_{n^2})$ via raster-scan ordering. In unified vision-language models (Wu et al., 2025; Chen et al., 2025; Team, 2024), visual and text tokens share the same vocabulary and transformer backbone, enabling autoregressive generation:

$$t_i \sim p_\theta(t_i \mid \mathbf{c}, t_1, \ldots, t_{i-1}), \tag{1}$$

where $\mathbf{c}$ is the text prompt prefix. Generating an $n \times n$ image needs n^2 sequential forward passes.

3.2 Accelerating Inference via 1D Speculative Decoding

Autoregressive (AR) image generation loads billions of parameters from memory at every step yet produces only a single token per forward pass, creating a memory-bandwidth bottleneck (Pope et al., 2023) known as the memory wall. Speculative decoding accelerates generation by first drafting candidate tokens and verifying them in a single forward pass, mitigating the memory wall by significantly reducing the number of times the large language model (LLM) parameters must be reloaded to predict sequential tokens.

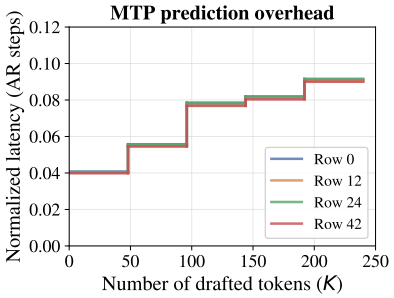


Figure 4: MTP drafting cost of SSD, normalized by one AR step on Lumina-mGPT-7B (48×48 grid). Even at 240 drafted tokens, overhead stays below 0.1 AR steps.

Multi-token prediction (MTP) (Gloeckle et al., 2024; Liu et al., 2024a; 2025) is a widely used drafting method that adds K lightweight prediction heads to the backbone. These heads, which are vastly smaller than the backbone, predict the next K tokens simultaneously using the original head features with marginal additional computational cost. The drafted tokens are then verified simultaneously by the backbone, requiring one parameter load instead of K sequential loads. As shown in Figure 4, MTP drafting heads are exceptionally lightweight: drafting up to 240 tokens costs less than 0.1 AR steps.

3.3 Spatially Speculative Decoding

Standard MTP drafting is inherently one-dimensional, sequentially predicting the immediate next tokens. While suitable for naturally 1D text sequences, 1D drafting along a raster-scan sequence is suboptimal for images because it ignores their two-dimensional structure. This limitation becomes severe for distant spatial neighbors. For instance, predicting the token directly below a given position in a row of width 48 requires a $+48$ offset head in the flattened sequence. Training such long horizon heads for every offset from $+1$ to $+48$ under standard MTP is inefficient, as it requires a separate head for each offset, and accuracy typically degrades with sequence distance.

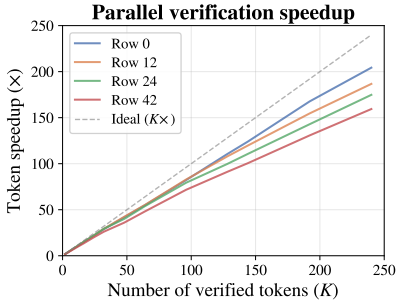


Figure 5: Token acceleration through parallel verification on Lumina-mGPT-7B. Wall-clock speedup over sequential AR scales near-linearly with the number of simultaneously verified tokens K, reaching $205\times$ at $K{=}240$.

However, we find that this degradation is an artifact of the raster scan order, not the underlying visual geometry. A token directly below a position is spatially adjacent despite its distance in the flattened sequence. Empirically, vertical heads at matching spatial offsets are just as accurate as their horizontal counterparts: a $+48$ vertical prediction matches a $+2$ horizontal prediction, and a $+96$ vertical prediction matches a $+3$ horizontal prediction (Figure 1). This observation suggests that MTP should be structured according to image geometry rather than sequence distance. Instead of training heads for arbitrary raster offsets, we train a small, localized set, such as $k_h = 3$ horizontal heads for offsets $(+1, +2, +3)$ and $k_v = 3$ vertical heads for offsets $(+48, +96, +144)$, bypassing the dense intermediate heads required by standard 1D-MTP.

Vertical prediction also offers a crucial advantage absent in horizontal prediction: column-wise independence. Along a raster-scan row, tokens must be generated sequentially because each token conditions on previous tokens in that same row. In contrast, vertical prediction can draft one token per column directly from the completed row above. Because these predictions do not depend on each other within the target row, they can be produced simultaneously, allowing an entire row of n tokens to be drafted in a single parallel step.

Our method, *Spatially Speculative Decoding* (SSD), factorizes 2D anticipation into two orthogonal 1D prediction streams. We first use k_h horizontal heads to draft along the raster-scan direction until the current row is completed. We then apply k_v vertical heads to draft up to k_v subsequent rows in parallel (Figure 2). The drafted rows are subsequently checked and corrected via *verification as auto correction*. By replacing token-by-token generation within each row with parallel row drafting, SSD reduces the number of backbone forward evaluations and alleviates the memory bottleneck of autoregressive decoding.

Draft Prediction in Latent Space Anticipating in discrete token space demands the exact codebook entry out of tens of thousands of candidates. Visual codebooks induce flat distributions over this vast label set, and exact-match acceptance rates fall below 5% (So et al., 2025). We empirically find that predicting in the continuous latent space—specifically, targeting the last transformer layer's hidden state before the final RMSNorm (Zhang & Sennrich, 2019) (see Appendix A.9 for ablations)—produces better MTP accuracy. A lightweight predictor f_ϕ takes as input the hidden state $\mathbf{h}_{y,x}$ concatenated with the token embedding $\mathbf{e}_{y,x}$ (to resolve sampling non-determinism (Li et al., 2024a)) and predicts the hidden state at offset δ:

$$\hat{\mathbf{h}}_{y+\delta,x} = f_\phi\big([\mathbf{h}_{y,x};\ \mathbf{e}_{y,x}]\big), \quad f_\phi(\mathbf{z}) = \mathbf{W}_0\mathbf{z} + \mathrm{SwiGLU}\big(\mathrm{RMSNorm}(\mathbf{W}_0\mathbf{z})\big), \tag{2}$$

where $\mathrm{SwiGLU}(\tilde{\mathbf{z}}) = \mathbf{W}_2\big(\sigma(\mathbf{W}_1\tilde{\mathbf{z}}) \odot \mathbf{W}_3\tilde{\mathbf{z}}\big)$ (Shazeer, 2020) with σ denoting SiLU (Ramachandran et al., 2017), $\mathbf{W}_0 \in \mathbb{R}^{d\times 2d}$, $\mathbf{W}_1, \mathbf{W}_3 \in \mathbb{R}^{m\times d}$, $\mathbf{W}_2 \in \mathbb{R}^{d\times m}$ are bias-free projections. Each offset δ and spatial direction has its own predictor. Draft tokens are decoded through the base model's existing output layers: $\hat{t}_{y+\delta,x} = \arg\max_v \texttt{GenHead}(\texttt{RMSNorm}(\hat{\mathbf{h}}_{y+\delta,x}))_v$. Training minimizes a smooth L1 loss (Girshick, 2015) against ground-truth hidden states from the frozen backbone:

$$\mathcal{L}_\delta = \frac{1}{|\mathcal{P}_\delta|} \sum_{(y,x)\in\mathcal{P}_\delta} \mathrm{SmoothL1}\big(\hat{\mathbf{h}}_{y+\delta,x},\ \mathbf{h}_{y+\delta,x}\big), \tag{3}$$

where $\mathcal{P}_\delta$ is the set of valid source positions. The pretrained model remains entirely unmodified.

Verification as Auto Correction. Once future tokens are drafted, we pass the entire drafted block through the frozen backbone in parallel for verification and correction. Because spatial drafting predicts one or more complete rows at a time, strict speculative verification that rolls back all positions after the first mismatch would largely eliminate the benefit of row-parallel drafting. We instead adopt a relaxed correction rule that repairs mismatched positions in parallel over a fixed number of verification rounds. This type of non-exact distribution preserving acceleration algorithm has been studied in prior work He et al. (2025a); Jang et al. (2025); Li et al. (2026b).

Table 1: Text-to-image generation results on **DPG-Bench**. For each model, we compare our SSD with original model's autoregressive inference, and speculative decoding baselines: 1D-MTP, SJD (Teng et al., 2025), GSD (So et al., 2025), LANTERN (Jang et al., 2025), and ZipAR (He et al., 2025a). Blue rows indicate our method. Our approach establishes a stronger speed–quality Pareto frontier, matching or exceeding prior acceleration baselines in generation quality at substantially higher speed.

Model	Latency (↓)	Acceleration (↑)		Global	Entity	Attribute	Relation	Other	Average (↑)
		Step	Latency						
Lumina-mGPT (7B)	96.23s	1.00×	1.00×	83.28	82.79	80.89	88.11	60.98	76.30
+ 1D-MTP	49.91s	2.23×	1.93×	76.78	73.11	74.61	83.67	56.73	63.20
+ SJD	61.08s	2.23×	1.58×	82.30	81.88	79.06	86.96	58.78	74.03
+ GSD	38.23s	2.89×	2.52×	83.89	81.18	79.38	87.54	59.60	73.37
+ LANTERN	59.00s	2.00×	1.63×	82.98	82.31	79.40	88.63	57.60	74.45
+ ZipAR	36.45s	2.94×	2.64×	83.80	82.70	80.81	88.05	61.60	74.62
+ SSD (Ours)	**13.09s**	**8.40×**	**7.35×**	81.73	82.60	79.77	88.70	58.54	74.57
Janus-Pro (7B)	7.87s	1.00×	1.00×	81.06	90.32	87.47	93.41	79.27	84.23
+ 1D-MTP	3.94s	2.48×	2.00×	79.69	87.80	86.00	92.96	72.36	80.98
+ SJD	5.25s	1.81×	1.50×	83.33	89.44	87.36	93.83	79.59	84.01
+ GSD	3.31s	3.07×	2.38×	81.46	89.43	87.55	93.38	78.80	82.97
+ LANTERN	9.32s	1.17×	0.84×	82.07	89.73	87.47	93.31	82.40	84.07
+ ZipAR	3.04s	2.76×	2.59×	79.94	89.60	87.65	93.58	80.80	83.78
+ SSD (Ours)	**1.38s**	**7.20×**	**5.70×**	82.61	88.90	87.33	93.46	77.64	83.40
Emu3 (8B)	281.87s	1.00×	1.00×	86.02	85.69	85.26	91.89	64.23	78.69
+ 1D-MTP	144.26s	2.39×	1.95×	72.05	67.42	62.87	85.27	46.94	53.11
+ SJD	260.41s	2.24×	1.08×	84.83	87.00	85.28	91.55	66.94	78.38
+ GSD	97.07s	3.46×	2.90×	82.97	85.73	82.58	91.80	65.85	77.57
+ LANTERN	182.50s	2.16×	1.54×	82.66	84.00	81.47	89.33	71.95	76.12
+ ZipAR	79.71s	5.41×	3.54×	83.75	81.85	82.33	89.73	67.35	73.36
+ SSD (Ours)	**25.55s**	**14.34×**	**11.03×**	81.48	84.31	81.59	90.35	58.30	76.03

For each drafted position k, the draft distribution q_k produces a candidate token $\hat{t}_k = \arg\max_v q_k(v)$. During the subsequent verification-and-correction forward pass, the LLM backbone produces an updated distribution p_k. We consider three strategies for selecting the final token at each drafted position: **(A) Greedy selection**, which directly takes $\arg\max_v p_k(v)$; **(B) Sampling**, which samples a token from p_k; and **(C) Acceptance–rejection sampling**, our default strategy. Specifically, we accept the drafted token $\hat{t}_k$ with probability $\alpha(\hat{t}_k) = \min\big(p_k(\hat{t}_k)/q_k(\hat{t}_k), 1\big)$. If the draft is rejected, we instead sample from the residual distribution $\tilde{p}_k(v) = [p_k(v) - q_k(v)]_+ / \sum_{v'}[p_k(v') - q_k(v')]_+$, where $[x]_+ = \max(0, x)$. This correction is applied independently to all drafted positions in parallel. We compare these three strategies in Table 3, where acceptance-rejection sampling achieves the best performance.

We repeat this verify-and-correct procedure for a fixed number of rounds r. After each round, rejected positions are replaced by samples from the residual distribution above, and the backbone is re-evaluated on the updated block. For a rejected position, the previous-round backbone distribution becomes its reference distribution in the next round, while accepted positions retain their current reference distributions. Increasing r progressively refines the drafted block toward the autoregressive model at the cost of additional latency, making r an explicit knob for controlling the speed–quality trade-off. After the final round, one additional forward pass commits the corrected block and updates the KV cache for subsequent generation.

4 Experiments

Models. We evaluate our method on three state-of-the-art autoregressive visual generation models: Janus-Pro-7B (Chen et al., 2025), Lumina-mGPT-7B (Liu et al., 2024b), and Emu3-8B (Wang et al., 2024), which produce token grids of 24×24 (576), 48×48 (2,304), and 90×90 (8,100 tokens), respectively. These sequence lengths span over an order of magnitude, providing a rigorous testbed to evaluate the generality of our approach. All pretrained backbones remain frozen; only the lightweight spatial draft predictors are trained.

Training. Each spatial draft predictor is trained via self-distillation (Hinton et al., 2015) using Midjourney prompts (Huggingface, 2024). To construct the training data, we first generate paired images for each input text prompt by performing inference with the original model. We then train

Table 2: Text-to-image generation results on **GenEval**. For each model, we compare our SSD with the autoregressive (AR) baseline. **Ours** denotes a configuration that achieves a competitive trade-off between generation quality and inference acceleration. Blue rows indicate our method. Latencies are measured on GenEval prompts for Janus-Pro and Emu3; for Lumina-mGPT we report the DPG-Bench measurement of the same configuration. Our approach achieves up to $11.24\times$ latency speedup while retaining competitive generation quality.

Model	Latency (↓)	Acceleration (↑)		Single	Two	Count.	Colors	Pos.	Col. Attr.	Average (↑)
		Step	Latency							
Lumina-mGPT (7B)	96.23s	1.00×	1.00×	0.99	0.75	0.20	0.80	0.13	0.16	0.50
+ SSD (Ours)	**13.09s**	**8.40×**	**7.35×**	0.97	0.62	0.19	0.72	0.11	0.13	0.46
Janus-Pro (7B)	7.72s	1.00×	1.00×	0.99	0.86	0.53	0.87	0.73	0.62	0.77
+ SSD (Ours)	**1.45s**	**7.20×**	**5.32×**	0.99	0.81	0.45	0.90	0.69	0.57	0.73
Emu3 (8B)	281.06s	1.00×	1.00×	0.95	0.50	0.28	0.77	0.10	0.13	0.45
+ SSD (Ours)	**25.00s**	**14.34×**	**11.24×**	0.86	0.38	0.20	0.72	0.10	0.11	0.40

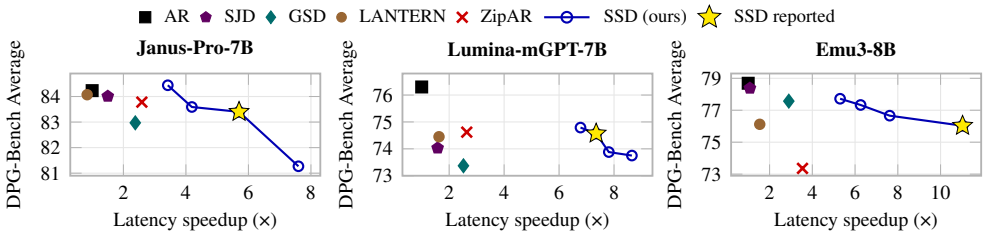


Figure 6: **SSD speed–quality trade-off curves.** Varying the verification budget controls the trade-off between generation quality and latency speedup. Each panel compares SSD with decoding baselines using the same backbone as in Table 1. Yellow stars mark representative operating points that balance quality and speed, corresponding to the settings reported in Tables 1 and 2. The top-right region is ideal; across backbones, SSD consistently achieves comparable generation quality at substantially higher speedups.

the lightweight spatial MTP heads on those data. For number of training data, we use: 60,000 for Janus-Pro, 20,000 for Lumina-mGPT, and 5,000 for Emu3. Following the classifier-free guidance protocol (Ho & Salimans, 2022), we randomly drop 10% of the text conditioning during training to enable unconditional generation. The five horizontal and vertical heads are trained independently using identical hyperparameters.

Metrics. To evaluate generation quality, we report results on two widely used text-to-image benchmarks. *DPG-Bench* (Hu et al., 2024) evaluates text-image semantic alignment under dense, compositionally rich prompts, providing a fine-grained assessment of how faithfully a model renders complex descriptions. *GenEval* (Ghosh et al., 2023) measures compositional generation fidelity through object-focused evaluation, testing whether models correctly produce specified objects, attributes, and spatial layouts. For inference speed, we report efficiency across three key metrics that align with Table 1, 2. Under **Acceleration (↑) Step**, we report the reduction factor in the NFE of the backbone. Because the memory wall bottleneck is driven by the repeated loading of model parameters, reducing these forward steps directly minimizes memory I/O and yields actual time savings. We also report the absolute wall-clock **Latency (↓)** in seconds, measured on an RTX Blackwell 6000 Pro GPU. Finally, **Acceleration (↑) Latency** quantifies the relative wall-clock speedup achieved by our method compared to the baseline autoregressive approach.

Baselines. We compare against six decoding methods, all of which keep the pretrained backbone frozen. **AR** denotes standard autoregressive decoding with the unmodified pretrained model. **1D-MTP** uses only the horizontal heads from our framework with the same speculative decoding procedure and generation hyperparameters as the horizontal component of SSD, isolating the contribution of vertical drafting. **SJD** (Teng et al., 2025) is a Jacobi-based parallel decoding method that iteratively refines multiple tokens along the 1D raster-scan sequence. **GSD** (So et al., 2025) groups interchangeable visual tokens in the codebook and accepts a drafted token whenever it falls into the same group as the target token. **LANTERN** (Jang et al., 2025) relaxes the acceptance test using latent-space similarity

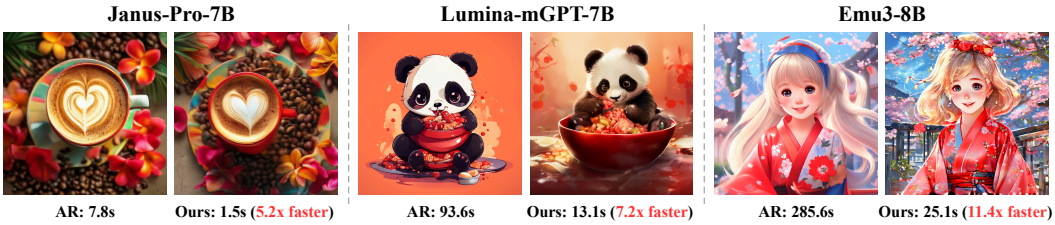


Figure 7: **Qualitative results.** Side-by-side comparison of AR baseline and SSD across three models. Our method yields up to 11.03× speedup while preserving strong visual quality.

Table 3: To assess the importance of verification as auto correction, we compare three draft-update strategies on Lumina-mGPT-7B (**DPG-Bench**) under the fixed b=5, i=4 schedule. All methods start from the same SSD draft and use the same forward-step budget, differing only in how they verify and correct it. Argmax and direct sampling fail to preserve generation quality, while our acceptance-rejection as auto correction achieves substantially better quality.

Method	Steps (↓)	DPG-Bench Average (↑)
(A) Greedy Selection (Argmax)	280	66.41
(B) Sampling	280	66.16
(C) Acceptance-rejection sampling (Ours)	280	**74.57**

between visual tokens, trading distributional exactness for higher acceptance rates. **ZipAR** (He et al., 2025a) is a training-free spatial scheme that decodes tokens from adjacent rows in parallel as soon as their local context is available.

SSD Configuration. All three models share the same horizontal drafting setup: 5 tokens per row with 1 verification round. For vertical drafting, Lumina-mGPT drafts k_v=2 rows per iteration with staged verification—5 base rounds on both rows followed by 4 incremental rounds on the trailing row (b=5, i=4)—as its 48×48 grid benefits from multi-row drafting (Appendix A.10). Janus-Pro and Emu3 draft a single row (k_v=1) with r=2 and r=5 verification rounds, respectively; on these grids the per-step cost of processing additional rows outweighs the step reduction from multi-row drafting.

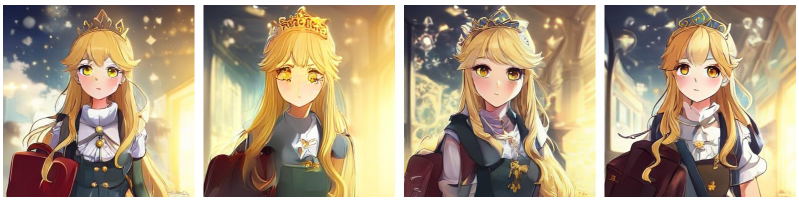
4.1 Main Results

On DPG-Bench (Table 1), ZipAR is the fastest prior baseline on all three backbones, achieving 2.59×, 2.64×, and 3.54× latency speedups on Janus-Pro, Lumina-mGPT, and Emu3, respectively. Our SSD pushes these speedups to 5.70×, 7.35×, and 11.03×, while closely matching ZipAR's quality on Janus-Pro (83.40 vs. 83.78) and Lumina-mGPT (74.57 vs. 74.62), and improving it on Emu3 (76.03 vs. 73.36). The same operating points retain competitive quality on GenEval (Table 2). Figure 6 further traces SSD's speed–quality frontier, showing that additional verification improves quality while retaining substantial speedups. Larger token grids provide more opportunities for row-parallel drafting and verification, leading to greater acceleration. Qualitative results are shown in Figure 7, with additional baseline comparisons in Appendix A.11.

4.2 Analysis

Importance of Verification as Auto Correction. We compare three strategies for updating the same SSD draft on DPG-Bench with Lumina-mGPT-7B (Table 3): (1) taking the argmax of the backbone distribution produced by each forward step conditioned on the current draft, (2) sampling from the same distribution, and (3) our method, which uses a probability-based accept/reject step to decide whether to keep each drafted token or resample it. All strategies use the same number of backbone forward steps. Verification as auto correction yields substantially better generation quality.

The Effect of Number of Correct Rounds. To examine the quality–speed tradeoff, we vary the number of per-row verification and correction rounds r (Table 4, Figure 8). The unverified draft (r=0) preserves coarse spatial layout but lacks fine detail. Each additional round progressively recovers local coherence, with quality improving sharply from r=0 (71.65) to r=2 (83.40). At r=2, the output is visually indistinguishable from the AR baseline while retaining 5.70× latency acceleration.



AR $r = 0$ $r = 1$ $r = 2$

Figure 8: Qualitative comparison of Janus-Pro-7B outputs under different vertical verification and correction rounds (r). The unverified draft (r=0) preserves coarse spatial layout, indicating that the spatial draft prediction acquires sufficient vertical coherence during training. Successive verification rounds progressively recover fine-grained detail and local coherence; by r=2 the output is visually indistinguishable from the autoregressive baseline.

Table 4: Ablation on vertical verification rounds (r) on Janus-Pro-7B (**DPG-Bench**). r denotes the number of verify-and-correct iterations applied to each vertically drafted row. The draft alone (r=0) produces coherent outputs, indicating that the spatial draft prediction captures sufficient spatial structure to serve as a reliable initialization. Subsequent verification rounds monotonically improve fidelity, recovering AR-level quality by r=2 while preserving substantial acceleration.

r	Steps (↓)	Latency (↓)	Acceleration (↑) Steps	Acceleration (↑) Latency	DPG-Bench Average (↑)
AR	576	7.87s	1.00×	1.00×	84.23
0	34	**0.63s**	**16.90×**	**12.50×**	71.65
1	57	1.03s	10.10×	7.60×	81.27
2	80	1.38s	7.20×	5.70×	**83.40**

Training samples	DPG-Bench (↑)
5K	54.67
10K	61.25
20K	64.20
30K	65.50
60K	**71.65**

Table 5: Effect of training data size on spatial draft prediction draft quality (Janus-Pro-7B, **DPG-Bench**, r=0). With only 5K samples the predictor already produces coherent drafts, and performance improves steadily through 60K. Blue indicates our default.

Vertical Lookahead Depth of Multi-Row Anticipation. To determine the optimal number of rows drafted per iteration, we vary the vertical lookahead depth k_v with the staged schedule fixed at b=5, i=4 (Table 20 in Appendix A.10). Increasing k_v reduces the total step count, but with sharply diminishing returns: going from k_v=1 to k_v=2 removes 23 steps, whereas going from k_v=2 to k_v=3 removes only 8. Each forward pass meanwhile processes proportionally more tokens, so latency flattens out: the gap between k_v=2 and k_v=3 is 0.27s, a tenth of the gap between k_v=1 and k_v=2. Quality, in contrast, peaks at k_v=2 (74.57 versus 73.42 and 74.31). We therefore use k_v=2 as the default, giving up 2% latency relative to k_v=3 for the best generation quality.

Effect of Scaling Training Data. To investigate the impact of training data scale, we vary the number of ground-truth samples used to train the spatial draft prediction (Table 5), evaluated at r=0 to isolate draft quality. The single-layer predictor already produces coherent drafts from just 5K hidden states (54.67), confirming that the continuous latent space provides a strong inductive bias for learning. Scaling from 5K to 60K yields a continuous upward trajectory, improving quality from 54.67 to 71.65. This upward trend shows no indication of saturation, suggesting that exposure to a broader distribution of visual representations continues to sharpen the predictor's spatial anticipation.

5 Conclusion

We introduce Spatially Speculative Decoding (SSD), an approach that leverages the intrinsic two-dimensional locality of images to overcome the memory wall in autoregressive visual inference. Our results demonstrate that this geometric shift substantially accelerates state-of-the-art generative models while maintaining strong generation quality. Our findings suggest that abandoning the linear constraints of language modeling to respect natural visual geometry is a promising direction for real-time, high-resolution generative models.

AI USE STATEMENT

We used generative AI tools only to polish the writing of this paper (grammar and readability). We did not use them for any other task requiring disclosure under the ICLR 2027 AI Policy. All AI-suggested edits were reviewed sentence by sentence by the authors, who take full responsibility for the content.

ETHICS STATEMENT

SSD accelerates inference of frozen pretrained image generators and adds no new generative capability; it inherits the biases and misuse risks of the underlying models, whose safeguards should remain in place (Appendix A.13). All datasets and benchmarks are publicly available and used under their licenses. The human evaluation (Appendix A.4) involved volunteers comparing unlabeled pairs of generated images; no personal information was collected.

REPRODUCIBILITY STATEMENT

The method is described in Sections 3.1–3.3 and the experimental setup in Section 4. All training and inference hyperparameters are listed in Table 6 (Appendix A.1), configuration selection on held-out prompts is detailed in Appendix A.7, and hardware is specified in Appendix A. We will release our code upon acceptance.

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 *Proceedings of the 41st International Conference on Machine Learning*, pp. 5209–5235, 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.

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.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 14060–14079, 2024.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36: 52132–52152, 2023.

Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 15706–15734, 2024.

Yefei He, Feng Chen, Yuanyu He, Shaoxuan He, Hong Zhou, Kaipeng Zhang, and Bohan Zhuang. Zipar: Parallel autoregressive image generation through spatial locality. In *International Conference on Machine Learning*, pp. 22368–22378. PMLR, 2025a.

Yefei He, Yuanyu He, Shaoxuan He, Feng Chen, Hong Zhou, Kaipeng Zhang, and Bohan Zhuang. Neighboring autoregressive modeling for efficient visual generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19000–19010, 2025b.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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

Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. Ella: Equip diffusion models with llm for enhanced semantic alignment. *arXiv preprint arXiv:2403.05135*, 2024.

Huggingface. Midjourney prompts dataset. `https://huggingface.co/datasets/vivym/midjourney-prompts`, 2024.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Doohyuk Jang, Sihwan Park, June Yong Yang, Yeonsung Jung, Jihun Yun, Souvik Kundu, Sung-Yub Kim, and Eunho Yang. Lantern: Accelerating visual autoregressive models with relaxed speculative decoding. In *The Thirteenth International Conference on Learning Representations*, 2025.

Siqi Kou, Lanxiang Hu, Zhezhi He, Zhijie Deng, and Hao Zhang. Cllms: consistency large language models. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 25426–25440, 2024.

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

Haopeng Li, Jinyue Yang, Guoqi Li, and Huan Wang. Autoregressive image generation with randomized parallel decoding. In *The Fourteenth International Conference on Learning Representations*, 2026a.

Xingyao Li, Fengzhuo Zhang, Cunxiao Du, and Hui Ji. Annealed relaxation of speculative decoding for faster autoregressive image generation. *arXiv preprint arXiv:2601.09212*, 2026b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 28935–28948, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7421–7432, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*, 2025.

Boya Liao, Ying Li, Siyong Jian, and Huan Wang. Parallel jacobi decoding for fast autoregressive image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9008–9018, 2026.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

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*, 2024b.

Xiaohao Liu, Xiaobo Xia, Weixiang Zhao, Manyi Zhang, Xianzhi Yu, Xiu Su, Shuo Yang, See-Kiong Ng, and Tat-Seng Chua. L-mtp: Leap multi-token prediction beyond adjacent context for large language models. *arXiv preprint arXiv:2505.17505*, 2025.

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 *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pp. 932–949, 2024.

Elia Peruzzo, Guillaume Sautière, and Amirhossein Habibian. Multi-scale local speculative decoding for image generation. *arXiv preprint arXiv:2601.05149*, 2026.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624, 2023.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

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. Pmlr, 2021.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodolà. Accelerating transformer inference for translation via parallel decoding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12336–12355, 2023.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

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

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*, 2024a.

Quan Sun, Qiying Yu, Yufeng Cui, Fan Zhang, Xiaosong Zhang, Yueze Wang, Hongcheng Gao, Jingjing Liu, Tiejun Huang, and Xinlong Wang. Emu: Generative pretraining in multimodality. In *The Twelfth International Conference on Learning Representations*, 2024b.

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

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 *The Thirteenth International Conference on Learning Representations*, 2025.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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.

Yuqing Wang, Shuhuai Ren, Zhijie Lin, Yujin Han, Haoyuan Guo, Zhenheng Yang, Difan Zou, Jiashi Feng, and Xihui Liu. Parallelized autoregressive visual generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12955–12965, 2025.

Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, et al. Janus: Decoupling visual encoding for unified multimodal understanding and generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 12966–12977, 2025.

Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

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.

Lijun Yu, Jose Lezama, Nitesh Bharadwaj Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G Hauptmann, et al. Language model beats diffusion-tokenizer is key to visual generation. In *The Twelfth International Conference on Learning Representations*, 2024.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models. *Transactions on Machine Learning Research*, 2024, 2024.

Zhuoyang Zhang, Luke J Huang, Chengyue Wu, Shang Yang, Kelly Peng, Yao Lu, and Song Han. Locality-aware parallel decoding for efficient autoregressive image generation. *arXiv preprint arXiv:2507.01957*, 2025.

Junkang Zhou, Yefei He, Feng Chen, Weijie Wang, and Bohan Zhuang. Flashar: Efficient post-training acceleration for autoregressive image generation. *arXiv preprint arXiv:2605.09430*, 2026.

A Technical appendices and supplementary material

This supplementary material provides additional details that complement the main text. All experiments are carried out on NVIDIA RTX PRO 6000 Blackwell GPUs.

A.1 Training and Inference Details

See Table 6.

Table 6: Training and inference configuration for the spatial drafting heads. The learning-rate schedule and the optimizer are identical for all three backbones.

	Janus-Pro-7B	**Lumina-mGPT-7B**	**Emu3-8B**
Token grid	24×24	48×48	90×90
Params per head (M)	134.2	134.2	134.2
Training			
Training prompts	60,000	20,000	5,000
Batch size / GPU	128	32	8
GPUs	4	4	4
Epochs	3	3	3
Gradient accumulation	1	1	1
CFG drop rate	10%	10%	10%
Learning rate	$1\times10^{-4}\rightarrow1\times10^{-5}$, 20 warmup steps		
Optimizer	AdamW (β_1=0.9, β_2=0.95, wd=0.01)		
Training cost (GPU-h)	3.0	5.6	3.3
Inference			
Temperature	1.0	1.0	1.0
CFG weight	5.0	4.0	3.0
Horizontal heads k_h	5	5	5
Horizontal verify rounds	1	1	1
Vertical heads k_v	1	2	1
Vertical verify rounds	r=2	b=5, i=4	r=5

Table 7: Speed–quality trade-off on **DPG-Bench** as a function of the verification budget. For Lumina-mGPT the schedule uses b verification rounds on the first row of each block and i further rounds on each subsequent row. ZipAR is the fastest baseline on all three backbones. Blue rows indicate the default configuration reported in Table 1.

Method	Steps (↓)	Latency (↓)	Acceleration (↑)		DPG-Bench Average (↑)
			Step	Latency	
Janus-Pro-7B					
AR (baseline)	576	7.87s	1.00×	1.00×	84.23
ZipAR (w=8)	209	3.04s	2.76×	2.59×	83.78
SSD (r=0)	34	0.63s	16.90×	12.50×	71.65
SSD (r=1)	57	1.03s	10.10×	7.60×	81.27
SSD (r=2)	80	**1.38s**	**7.20×**	**5.70×**	83.40
SSD (r=3)	103	1.88s	5.59×	4.19×	83.59
SSD (r=4)	126	2.30s	4.57×	3.42×	**84.44**
Lumina-mGPT-7B					
AR (baseline)	2,352	96.23s	1.00×	1.00×	76.30
ZipAR (w=16)	801	36.45s	2.94×	2.64×	74.62
SSD (b=5, i=1)	211	**9.97s**	**11.15×**	**9.65×**	72.22
SSD (b=5, i=2)	234	11.12s	10.05×	8.65×	73.75
SSD (b=5, i=3)	257	12.32s	9.15×	7.81×	73.88
SSD (b=5, i=4)	280	13.09s	8.40×	7.35×	74.57
SSD (b=5, i=5)	303	14.19s	7.76×	6.78×	**74.79**
Emu3-8B					
AR (baseline)	8,190	281.87s	1.00×	1.00×	78.69
ZipAR (w=16)	1,515	79.71s	5.41×	3.54×	73.36
SSD (r=5)	571	**25.55s**	**14.34×**	**11.03×**	76.03
SSD (r=8)	838	37.00s	9.77×	7.62×	76.66
SSD (r=10)	1,016	44.99s	8.06×	6.26×	77.33
SSD (r=12)	1,194	53.20s	6.86×	5.30×	**77.72**

Table 8: FID on COCO2017-val 5K (Janus-Pro-7B) as a function of verification rounds r. Blue indicates our default configuration.

Method	FID (↓)	Latency (↓)	Acceleration (↑)
AR (baseline)	29.19	7.87s	1.00×
SSD (r=0)	59.19	**0.63s**	**12.49×**
SSD (r=1)	31.26	1.03s	7.64×
SSD (r=2)	28.95	1.38s	5.70×
SSD (r=3)	**28.70**	1.85s	4.25×

A.2 Speed–Quality Trade-off

The number of verification rounds provides an explicit control over the speed–quality trade-off. Table 7 sweeps this budget across all three backbones, with ZipAR as the fastest baseline. Increasing the verification budget consistently improves quality, with diminishing returns at larger budgets: Janus-Pro reaches 84.44 at r=4, slightly above AR (84.23); Emu3 improves from 76.03 at r=5 to 77.72 at r=12, approaching AR (78.69); and Lumina-mGPT gains little beyond i=4. Across all three backbones, SSD provides an operating point that is both faster and higher-quality than ZipAR while remaining faster throughout the sweep.

A.3 Perceptual Fidelity

DPG-Bench and GenEval primarily measure text–image alignment rather than perceptual quality. We therefore additionally evaluate FID on COCO2017-val 5K with Janus-Pro-7B (Table 8). Verification rapidly closes the gap from the unverified draft: at r=2, SSD reaches 28.95 FID while retaining a 5.70× latency speedup. Over five random seeds (Table 9), SSD obtains a clean-FID of 28.89 ± 0.24 versus 29.13 ± 0.08 for AR. The difference of 0.25 has a 95% confidence interval of $[-0.04, +0.54]$ (Welch's p=0.08), indicating no statistically significant difference in FID.

Table 9: clean-FID on COCO2017-val 5K (Janus-Pro-7B) over five random seeds, reported as mean ± s.d. The difference in favour of SSD is $+0.25$ FID with a 95% confidence interval of $[-0.04, +0.54]$ (Welch's two-sample t-test, p=0.08), i.e. the two are statistically indistinguishable.

Method	clean-FID (↓)	Latency (↓)	Acceleration (↑)
AR (baseline)	29.13 ± 0.08	7.87s	$1.00\times$
SSD (r=2)	$\mathbf{28.89 \pm 0.24}$	**1.38s**	$\mathbf{5.70}\times$

Table 10: Human pairwise preference on 50 COCO2017-val prompts (Janus-Pro-7B), reported as mean ± s.d. across seven evaluators.

Method	Win rate (↑)	Latency speedup (↑)
AR (baseline)	$54.9 \pm 7.0\%$	$1.00\times$
SSD (r=2)	$45.1 \pm 7.0\%$	$\mathbf{5.70}\times$

A.4 Human Evaluation

Clean-FID compares distributions rather than individual images, and therefore cannot tell whether a human observer can distinguish a given SSD sample from its AR counterpart. We therefore conduct a pairwise preference study on 50 randomly sampled COCO2017-val prompts. For each prompt, Janus-Pro-7B generates one image with AR decoding and one with SSD (r=2) under the same seed, and seven evaluators compare the pair in randomized order without method labels; Figure 9 shows the evaluation interface and two example pairs. As shown in Table 10, AR and SSD receive preference rates of $54.9 \pm 7.0\%$ and $45.1 \pm 7.0\%$, respectively. Evaluator-level variation spans the 50% indifference point, indicating no clear perceptual preference between the two methods, while SSD generates images $5.70\times$ faster.

A.5 Characterizing the Deviation from Autoregressive Sampling

SSD verifies each row in parallel using a context that may still contain drafted tokens, it does not preserve the exact autoregressive distribution. We quantify this deviation on Janus-Pro-7B using K=100 randomly sampled COCO2017-val prompts and N=10 random seeds per prompt for both AR and SSD (r=2). For each prompt, we estimate the mean μ and covariance Σ of the generated images in Janus-Pro's visual feature space and pool the resulting statistics across prompts (Table 11).

The squared distance between the AR and SSD centers is 0.0212, corresponding to 0.70 of the AR conditional standard deviation, while the feature-space spread increases by 25% ($\mathrm{tr}\,\Sigma_{\mathrm{AR}} = 0.0428$ versus $\mathrm{tr}\,\Sigma_{\mathrm{SSD}} = 0.0533$). Thus, the center shift induced by parallel verification is small relative to the natural seed-to-seed variability of AR generation.

A.6 Computational Cost of the Drafting Heads

We quantify the overhead of the drafting heads at the operating points reported in Table 1. Each head contains 134.2 M parameters (Table 6), but only a subset is active at each stage. The five horizontal heads add 0.671 B parameters and 1.34 GB of bf16 memory but are used only to complete the first row of each block. During the dominant vertical stage, SSD activates only one or two vertical heads, adding 0.134–0.268 B parameters (1.6–3.8% of the backbone), 0.27–0.54 GB of memory, and 1.5–2.0% per-step FLOPs overhead. Training the heads requires only 3.0–5.6 GPU-hours.

SSD reduces sequential forward evaluations rather than total arithmetic. Table 12 accounts for all backbone and verification passes, attention, drafting heads, activations, and KV-cache memory. Compared with AR, SSD performs 2.7–$7.7\times$ more total FLOPs and uses 2.4–4.3 GB more peak memory, yet achieves 5.70–$11.03\times$ lower wall-clock latency. This behavior is consistent with memory-bandwidth-bound decoding, where repeated backbone parameter loading dominates latency.

Figure 4 reports drafting cost, while Figure 5 shows the near-linear token acceleration enabled by parallel verification on Lumina-mGPT. Table 13 extends the normalized drafting and verification cost measurements to all three backbones. At the one-row operating point, drafting costs only 0.063, 0.040, and 0.098 AR steps on Janus-Pro, Lumina-mGPT, and Emu3, respectively, while verification

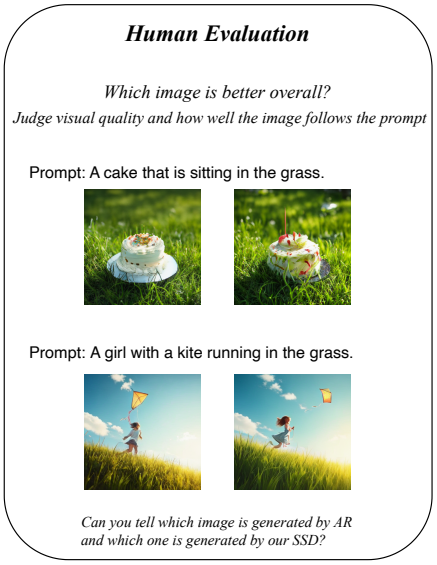


Figure 9: Two samples of the human preference study. Each screen shows a COCO2017-val prompt together with the AR and SSD (r=2) images generated from it under the same seed, placed side by side in randomized order and without method labels; the evaluator selects the image that is better overall in visual quality and prompt adherence.

Table 11: Deviation of SSD (r=2) from AR generation on Janus-Pro-7B, measured in Janus-Pro's visual feature space over K=100 COCO2017-val prompts $\times$ N=10 seeds.

Effect	Measured	Relative to AR
Location (center shift)	$\|\mu_{\mathrm{SSD}} - \mu_{\mathrm{AR}}\|^2 = 0.0212$	0.70 AR cond. s.d.
Scale (spread)	$\mathrm{tr}\,\Sigma_{\mathrm{AR}} = 0.0428,\ \mathrm{tr}\,\Sigma_{\mathrm{SSD}} = 0.0533$	+25%

costs 1.26, 1.20, and 1.22 AR steps. Generating the same rows autoregressively requires 24, 48, and 90 sequential steps. Thus, a row-level evaluation is only modestly more expensive than a single-token evaluation while replacing many sequential forward evaluations.

A.7 Configuration Selection

The configurations reported in Table 1 are selected on 100 held-out Midjourney prompts. Tables 14–16 report the full sweeps. For each backbone, we sweep the vertical lookahead v, base verification budget b, and incremental budget i, and select an reasonable point that balances acceptance rate and inference latency while visually inspecting samples at each candidate setting.

A.8 Generalization of the Drafting Heads

The drafting heads are trained on Midjourney prompts but evaluated on DPG-Bench and GenEval, which differ substantially in length, vocabulary, style, and compositional structure (Table 17). Roughly half of the DPG-Bench word types are unseen during training, the corpora share almost no 5-grams, and the evaluation benchmarks place substantially greater emphasis on spatial relations, attribute binding, and counting. Despite this shift, Emu3's heads, trained on only 5,000 prompts, reach 76.03 on DPG-Bench compared with 78.69 for AR while providing an 11.03$\times$ wall-clock speedup. Table 18 further shows that the same head architecture and training recipe transfer across three backbones with different resolutions, visual tokenizers, and a 14$\times$ range in tokens per image.

Table 12: End-to-end compute and memory per generated image, measured under identical optimized inference loops. Drafting-head rows are split into the horizontal heads, which run only on the first row of each block, and the vertical heads, which run on all remaining rows.

	Janus-Pro-7B		Lumina-mGPT-7B		Emu3-8B	
	AR	**SSD** (r=2)	**AR**	**SSD** (b=5, i=4)	**AR**	**SSD** (r=5)
Compute per image (TFLOPs)						
Backbone, all forward passes	16.6	43.8	65.5	499	255	1,521
Attention	0.224	0.646	3.08	24.3	35.9	216
Drafting heads, horizontal	—	0.012	—	0.025	—	0.048
Drafting heads, vertical	—	0.296	—	2.42	—	4.30
End-to-end	**16.8**	**44.8**	**68.6**	**526**	**291**	**1,742**
Memory (GB)						
Model weights	14.86	16.47	14.37	16.25	18.23	19.84
of which drafting heads	—	1.61	—	1.88	—	1.61
Activations + KV cache	1.33	2.12	2.61	2.79	4.36	7.02
Peak GPU memory	**16.18**	**18.59**	**16.98**	**19.04**	**22.59**	**26.86**
Wall-clock latency (s)	7.87	**1.38**	96.23	**13.09**	281.87	**25.55**

Table 13: Drafting and verification cost across all three backbones, normalized by one AR decoding step at the same generation position. K is a multiple of the grid width because drafting is row-parallel. Values are means over four generation positions with 50 timed repetitions each.

Model (grid)	Tokens K	Drafting (AR steps)	Verification (AR steps)
Janus-Pro-7B (24×24)	24	0.063	1.256
	48	0.091	1.263
	72	0.142	1.365
	96	0.155	1.407
	120	0.174	1.433
Lumina-mGPT-7B (48×48)	48	0.040	1.202
	96	0.055	1.220
	144	0.078	1.274
	192	0.081	1.300
	240	0.091	1.335
Emu3-8B (90×90)	90	0.098	1.218
	180	0.150	1.525
	270	0.236	2.071
	360	0.271	2.506
	450	0.337	2.571

A.9 Prediction Target Space

We vary the spatial draft head's input and target representations to find out the optimal feature to predict for SSD (Table 19). Operating in discrete token space ($\mathbf{e} \to \mathbf{e}$) collapses to 5.52—the flat codebook distribution provides too little supervisory signal, producing outputs devoid of recognizable structure (Figure 10). Switching to continuous hidden-state targets recovers strong draft quality, with pre-norm states ($\mathbf{h}_{\text{pre}}$) consistently surpassing their post-norm counterparts ($\mathbf{h}_{\text{post}}$), as the unnormalized representation retains a richer feature distribution. Appending the token embedding to the input further resolves the sampling ambiguity intrinsic to greedy decoding from hidden states (Li et al., 2024a), achieving the strongest average score of 71.65.

A.10 Multi-Row Anticipation Ablations

Verification Schedule. When anticipating multiple rows, jointly verifying all rows simultaneously causes inter-row correlations to destabilize convergence, whereas staged verification—committing the leading row before refining the trailing row—provides corrected context that yields consistently higher quality.

To study multi-row verification, we compare joint and staged schedules at the same total verification budget (b+i=9, Table 21). Joint verification applies all rounds to both drafted rows simultaneously, whereas staged verification first commits the leading row and then refines the trailing row against the corrected context. Under joint verification, the leading row remains unstable across rounds and the

Table 14: Configuration sweep on Janus-Pro-7B over 100 held-out Midjourney prompts. AR reference: 576 steps, 7.87 s. Accept rate is undefined at b=0 because no verification is performed. Blue indicates the deployed configuration.

Config	v	b	i	Steps (↓)	Latency (↓)	Speedup (↑)	Accept rate (↑)
v1b0i0	1	0	0	34	0.59s	13.42×	—
v1b1i0	1	1	0	57	0.98s	8.00×	0.1657
v1b2i0 (r=2)	1	2	0	80	1.38s	5.70×	0.4305
v1b3i0	1	3	0	103	1.88s	4.19×	0.5571
v1b4i0	1	4	0	126	2.30s	3.42×	0.6302
v1b5i0	1	5	0	149	2.77s	2.84×	0.6778
v2b2i1	2	2	1	69	1.19s	6.61×	0.3378
v2b2i2	2	2	2	80	1.39s	5.65×	0.4418
v3b2i1	3	2	1	65	1.12s	7.02×	0.2978
v3b2i2	3	2	2	80	1.39s	5.65×	0.4471

Table 15: Configuration sweep on Lumina-mGPT-7B over 100 held-out Midjourney prompts. AR reference: 2,352 steps, 96.23 s. Blue indicates the deployed configuration.

Config	v	b	i	Steps (↓)	Latency (↓)	Speedup (↑)	Accept rate (↑)
v1b0i0	1	0	0	68	3.21s	29.95×	—
v1b1i0	1	1	0	115	5.43s	17.71×	0.6894
v1b2i0	1	2	0	162	7.65s	12.57×	0.6636
v1b3i0	1	3	0	209	9.88s	9.74×	0.7027
v1b4i0	1	4	0	256	12.09s	7.96×	0.7364
v1b5i0	1	5	0	303	14.22s	6.77×	0.7605
v1b6i0	1	6	0	350	16.35s	5.89×	0.7846
v1b7i0	1	7	0	397	18.47s	5.21×	0.7970
v1b8i0	1	8	0	444	20.60s	4.67×	0.8069
v2b5i1	2	5	1	211	9.97s	9.65×	0.6758
v2b5i2	2	5	2	234	11.12s	8.65×	0.7159
v2b5i3	2	5	3	257	12.32s	7.81×	0.7441
v2b5i4 (b=5, i=4)	2	5	4	280	13.09s	7.35×	0.7645
v2b5i5	2	5	5	303	14.19s	6.78×	0.7745
v3b5i1	3	5	1	179	8.46s	11.38×	0.6263
v3b5i2	3	5	2	210	9.92s	9.70×	0.6979
v3b5i3	3	5	3	241	11.41s	8.43×	0.7369
v3b5i4	3	5	4	272	12.82s	7.51×	0.7605
v3b5i5	3	5	5	303	14.22s	6.77×	0.7771

trailing row oscillates with it, yielding a DPG-Bench score of 70.93. Staged verification reduces this coupling and improves the score to 74.57. The qualitative difference is also visible in Figure 11.

A.11 Extended Visualizations

Figure 12 provides extended qualitative comparisons among AR, SJD, 1D-MTP, and SSD across all three backbones.

A.12 Limitations

Despite SSD's efficacy, it has several limitations. First, its relaxed parallel correction is not exactness-preserving: later positions may be verified under contexts that still contain drafted tokens, SSD does not sample from the exact autoregressive distribution. Second, SSD is a finite-budget approximate decoder and does not guarantee convergence to an exact fixed point; increasing the verification budget instead provides an explicit trade-off between latency and agreement with AR.

A.13 Broader Impacts

SSD substantially reduces wall-clock latency and the number of sequential forward evaluations required for autoregressive image generation, improving inference throughput. As an inference-only acceleration method, SSD does not introduce new generative capabilities beyond those of the

Table 16: Configuration sweep on Emu3-8B over 100 held-out Midjourney prompts. AR reference: 8,190 steps, 278.33 s on the same held-out set. Blue indicates the deployed configuration. *Linearly interpolated: this configuration had no contention-free measurement window.

Config	v	b	i	Steps (↓)	Latency (↓)	Speedup (↑)	Accept rate (↑)
v1b0i0	1	0	0	126	5.30s	52.5×	—
v1b1i0	1	1	0	215	9.18s	30.3×	0.2067
v1b2i0	1	2	0	304	13.07s	21.3×	0.4913
v1b3i0	1	3	0	393	16.94s	16.4×	0.6585
v1b4i0	1	4	0	482	20.85s	13.4×	0.7212
v1b5i0 (r=5)	1	5	0	571	24.71s	11.3×	0.7512
v1b6i0	1	6	0	660	28.57s	9.74×	0.7739
v1b7i0	1	7	0	749	32.46s	8.58×	0.7906
v1b8i0	1	8	0	838	36.35s	7.66×	0.7991
v2b6i1	2	6	1	440	22.03s	12.63×	0.6958
v2b6i2	2	6	2	484	23.98s	11.61×	0.7392
v2b6i3	2	6	3	528	25.90s	10.75×	0.7634
v2b6i4	2	6	4	572	27.84s	10.00×	0.7769
v2b6i5	2	6	5	616	29.75s	9.35×	0.7856
v2b6i6	2	6	6	660	31.68s	8.78×	0.7950
v3b6i1	3	6	1	365	21.56s	12.91×	0.6606
v3b6i2	3	6	2	424	24.50s	11.36×	0.7205
v3b6i3	3	6	3	483	27.41s	10.16×	0.7573
v3b6i4	3	6	4	542	30.33s*	9.18×	0.7804
v3b6i5	3	6	5	601	33.27s	8.37×	0.7935
v3b6i6	3	6	6	660	36.16s	7.70×	0.8025

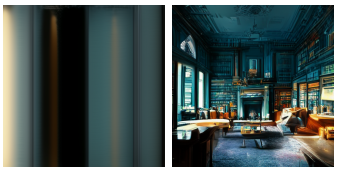
Table 17: Training prompts (Midjourney) versus evaluation benchmarks. Vocabulary and 5-gram statistics are computed against Emu3's 5,000 training prompts, the smallest of the three training sets.

	Training (Midjourney)	DPG-Bench	GenEval
Number of prompts	5K–60K	1,065	553
Median length (words)	14	65	8
Exact prompt overlap	—	0	0
Word types unseen in training	—	49.4%	29.3%
5-grams shared with training	—	0.35%	0.00%
Midjourney style modifiers	18.0%	7.1%	0.0%
Explicit spatial relations	4.5%	39.6%	18.1%
Colour–attribute binding	31.2%	72.0%	35.1%
Explicit counting	4.1%	5.6%	11.4%

underlying pretrained model. Safeguard mechanisms should therefore continue to evolve alongside improvements in generation throughput.

Table 18: The same head architecture and training recipe are applied to three backbones spanning different resolutions, visual tokenizers, and a $14\times$ range in tokens per image.

	Janus-Pro-7B	**Lumina-mGPT-7B**	**Emu3-8B**
Resolution	384^2	768^2	720^2
Visual tokenizer	VQ-16	Chameleon VQ	Emu3 VQ
Codebook size	16,384	8,192	32,768
Token grid	24×24	48×48	90×90
Tokens per image	576	2,304	8,100
Training images	60,000	20,000	5,000
DPG-Bench, AR $\rightarrow$ SSD	84.23 $\rightarrow$ 83.40	76.30 $\rightarrow$ 74.57	78.69 $\rightarrow$ 76.03
Wall-clock speedup	$5.70\times$	$7.35\times$	$11.03\times$



Token ($\mathbf{e} \rightarrow \mathbf{e}$) **Ours** ($\mathbf{h}_{\text{pre}} \oplus \mathbf{e} \rightarrow \mathbf{h}_{\text{pre}}$)

Figure 10: Samples generated by Janus-Pro-7B with different spatial draft prediction configurations at r=0 (no verification). The token-space head ($\mathbf{e} \rightarrow \mathbf{e}$) fails to reconstruct meaningful visual content, whereas our hidden-state head ($\mathbf{h}_{\text{pre}} \oplus \mathbf{e} \rightarrow \mathbf{h}_{\text{pre}}$) generates coherent images. Highlighting the importance of predicting the right target in our SSD.

Input	**Target**	**DPG-Bench Average** (↑)
$\mathbf{e}$	$\mathbf{e}$	5.52
$\mathbf{h}_{\text{pre}}$	$\mathbf{h}_{\text{pre}}$	69.07
$\mathbf{h}_{\text{post}}$	$\mathbf{h}_{\text{post}}$	45.26
$\mathbf{h}_{\text{pre}} \oplus \mathbf{e}$	$\mathbf{h}_{\text{pre}}$	71.65
$\mathbf{h}_{\text{post}} \oplus \mathbf{e}$	$\mathbf{h}_{\text{post}}$	41.57

Table 19: The impact of input and output representation for training spatial draft predictor. We train on Janus-Pro-7B and report the score on **DPG-Bench**. We evaluate without verification (r=0) to isolate draft quality. $\mathbf{h}_{\text{pre}}$/$\mathbf{h}_{\text{post}}$: last-layer hidden state before/after the final RMSNorm. $\mathbf{e}$: token embedding. $\oplus$: concatenation. Blue indicates our default. Predicting in discrete token space provides insufficient supervisory signal for the spatial draft prediction. Targeting pre-norm hidden states with token embedding concatenation provides the strongest learning signal.

Table 20: Vertical lookahead depth ablation on Lumina-mGPT-7B (**DPG-Bench**) with staged verification (b=5, i=4). k_v denotes the number of rows drafted in parallel per iteration. Increasing k_v reduces the number of forward passes but increases the token count per pass, so the step savings shrink quickly while the per-step cost keeps growing. k_v=2 achieves the best quality at close to the lowest latency, and is our default.

k_v	**Steps** (↓)	**Latency** (↓)	**Acceleration** (↑)		**DPG-Bench Average** (↑)
			Step	**Latency**	
AR	2,352	96.23s	$1.00\times$	$1.00\times$	76.30
1	303	14.22s	$7.76\times$	$6.77\times$	73.42
2	280	13.09s	$8.40\times$	$7.35\times$	**74.57**
3	272	**12.82s**	$\mathbf{8.65}\times$	$\mathbf{7.51}\times$	74.31



Joint (b=9, i=0) Staged (b=5, i=4) Joint (b=9, i=0) Staged (b=5, i=4)

Figure 11: Samples generated by Lumina-mGPT-7B under joint and staged verification schedules at matched total budget (b+i=9). Each pair compares joint verification (b=9, i=0), which processes both drafted rows simultaneously, against staged verification (b=5, i=4), which commits the first row before refining the second. Staged verification produces sharper details and more coherent spatial structure.

Table 21: Verification schedule ablation on Lumina-mGPT-7B (**DPG-Bench**) with vertical lookahead k_v=2. *Joint*: all b verification rounds operate on both drafted rows simultaneously. *Staged*: b base rounds on both rows, commit the first row, then i incremental rounds on the second row with the corrected first row as context. At matched total budget (b+i=9), staged verification outperforms joint (74.57 vs. 70.93) by providing corrected context before refining the trailing row.

Strategy	Schedule	Steps (↓)	Latency (↓)	Acceleration (↑)		DPG-Bench Average (↑)
				Step	Latency	
AR	–	2,352	96.23s	1.00×	1.00×	76.30
Joint	b=9, i=0	284	13.58s	8.28×	7.09×	70.93
Staged	b=5, i=1	211	**9.97s**	**11.15**×	**9.65**×	72.22
	b=5, i=2	234	11.12s	10.05×	8.65×	73.75
	b=5, i=3	257	12.32s	9.15×	7.81×	73.88
	b=5, i=4	280	13.09s	8.40×	7.35×	**74.57**

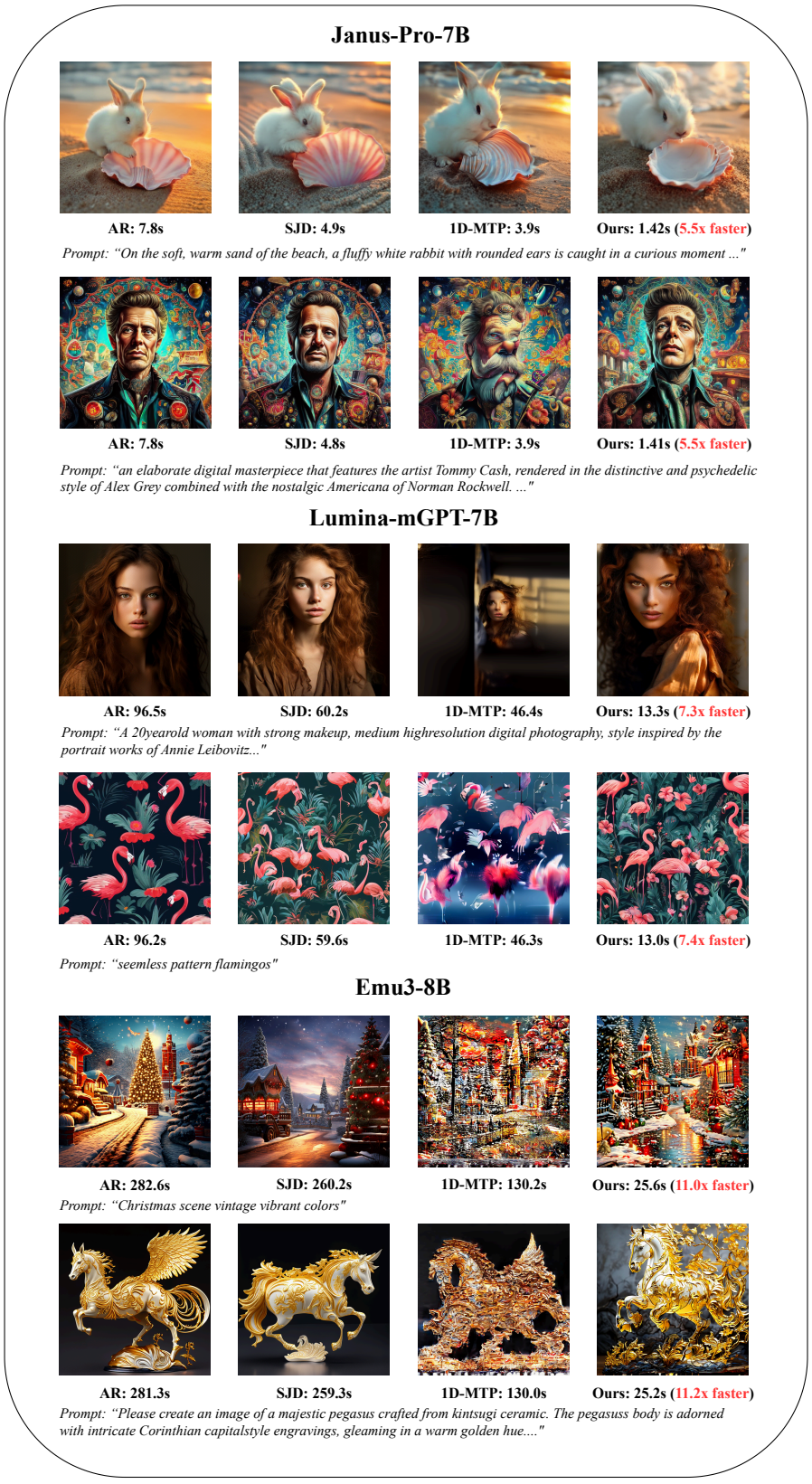


Figure 12: **Extended qualitative results.** Side-by-side comparisons of AR baseline, SJD, 1D-MTP, and SSD (Ours) across three models, demonstrating that our method achieves significant acceleration while maintaining high visual fidelity across diverse prompts.