DISPERSED ANY-ORDER SPECULATIVE DECODING FOR EXACT PARALLEL AUTOREGRESSIVE IMAGE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive image models generate one token per forward pass, which makes inference slow. Speculative decoding can speed up inference by generating multiple tokens per pass without changing the sampled distribution. The speedup depends on how much the drafter's distribution disagrees with the target's, and we show that this disagreement has two sources: 1) the drafter can be a different model than the target, and 2) it can condition on different tokens than the target since the drafter generates in parallel while the target conditions each draft on the ones before it. Recent self-speculative methods use the target model as its own drafter, which removes the first source. However, for left-to-right generation, since the drafted positions are adjacent and strongly dependent on one another, the second source of disagreement remains large. We find that disagreement decreases as distance between draft tokens increases, reducing the second source of disagreement. Based on this, we propose **D**ispersed Any-Order **S**peculative **D**ecoding (DSD), which uses an any-order model as its own drafter and verifier along an order whose consecutive positions are far apart. DSD requires no additional training and samples from the exact target model's distribution. On three any-order image models, DSD samples the exact distribution with up to $30\times$ fewer forward passes than sequential decoding, $4.6\times$ fewer than the best existing self-drafting method, and $5.5\times$ fewer than speculative decoding with a trained drafter, and it matches the quality of the models' lossy sampling with 5–$10\times$ fewer steps.



Figure 1: Left: agreement between the drafter's distribution q and the target's conditional p, i.e., the probability that a draft token is accepted, measured on ARPG-L. Rows vary the drafter's model and columns its conditioning (positions of tokens drafted together). When the drafter and target models are the same, q drafts the tokens of a step in parallel while p conditions each on the earlier ones (causal mask). Right: FDr^6 against target passes on RandAR-0.7B. Exact samplers lie on the model's distribution (dashed line), which lossy sampling reaches only at one token per step (square).

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.

1 INTRODUCTION

Autoregressive (AR) image models generate sequences one token at a time, each requiring a separate forward pass. The sequential nature of this process, along with the fact that each forward pass has low arithmetic intensity and is thus memory bound, makes AR inference slow (Pope et al., 2023).

Many methods have been proposed to speed up AR inference. Lossy methods do so by sampling several tokens independently, which approximates the joint distribution (Chang et al., 2022; Gloeckle et al., 2024). Any-order AR models allow lossy sampling by generating multiple tokens at arbitrary positions in parallel (Pang et al., 2025; Li et al., 2026; Zhang et al., 2026). While lossy methods can significantly speed up inference, they come at the cost of quality. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023), on the other hand, consists of sampling multiple tokens from a cheap drafter, scoring those draft tokens against the target model in a single pass, and accepting each draft token $\hat{x}$ with probability $\min(1, p(\hat{x})/q(\hat{x}))$, where p and q are the target and drafter distributions, respectively. The first rejected token is resampled from the residual distribution $\mathrm{norm}(\max(0, p-q))$. This speeds up inference since multiple tokens can be accepted at once (with the same step count as AR decoding at worst), while guaranteeing that the output is a sample from the exact distribution of the target model.

With speculative decoding, a token sampled from the draft distribution q is accepted with expected probability $\sum_x \min(q(x), p(x)) = 1 - \mathrm{TV}(q, p)$, where $\mathrm{TV}(q, p)$ is the total variation between the drafter's distribution and the target model's conditional distribution given the earlier draft tokens of that step (Leviathan et al., 2023; Sun et al., 2023; Yin et al., 2024). The speedup therefore depends on how closely the drafter distribution matches the target's. Their mismatch can arise from two sources (Figure 1, left). First, the drafter can be a different model from the target. Second, the drafter and target can condition on different information. For instance, a drafter that proposes all tokens of a step in parallel does not see the earlier draft tokens of that step, whereas the target does. Self-speculative decoding (or self-drafting) eliminates the first source of mismatch by using the target model itself as the drafter (Santilli et al., 2023; Teng et al., 2025; So et al., 2026). The remaining mismatch arises solely from differences in conditioning. Since, when self-drafting, the draft tokens within a step are proposed in parallel, each token is generated without conditioning on the earlier draft tokens from the same step. Consequently, the resulting generation-verification disagreement is governed by the dependence among the drafted positions. For a left-to-right AR model, these positions are necessarily contiguous, where dependence tends to be strong.

An any-order AR model, in contrast, can generate tokens at arbitrary positions. It can therefore be used for self-speculative decoding by drafting any set of positions in parallel and verifying them in one pass. The generation and verification order therefore becomes a design choice rather than being fixed by the left-to-right factorization. We observe that token dependence generally decreases as the distance between positions increases. Motivated by this observation, we propose Dispersed any-order Speculative Decoding (DSD), which uses an any-order model as both drafter and verifier and follows a generation order designed so that consecutive positions are far apart. DSD requires no additional training and preserves exactness: its outputs are distributed identically to those obtained by sequential sampling from the target model.

Our contributions are the following. **(i)** We perform an analysis of the disagreement between drafter and target in speculative decoding, showing that trained drafters significantly disagree with the target even when given the target's conditioning, whereas the disagreement of a self-drafting model can be reduced to the dependence between the drafted positions, which is controlled by the generation order. **(ii)** Based on the analysis, we propose DSD, a speculative decoding method for any-order models in which the model is its own drafter and the positions are drafted along a dispersed order. **(iii)** We perform experiments on three any-order image models. DSD uses up to $30\times$ fewer forward passes than sequential decoding while sampling from the same exact distribution. It uses 3–$5\times$ fewer passes than existing speculative decoding methods (Figure 1, right). On multi-representation Fréchet distances (Yang et al., 2026), lossy sampling needs 5.2–$10.1\times$ more steps than DSD to reach its quality.

2 BACKGROUND

Autoregressive generation. An autoregressive model over a sequence $x = (x_1, \ldots, x_N)$ of N discrete tokens, conditioned on c, factorizes the joint distribution as

$$p(x \mid c) \;=\; \prod_{t=1}^{N} p\big(x_t \,\big|\, x_{<t},\, c\big), \tag{1}$$

and samples it causally one token at a time, each from its conditional given the already generated tokens. Generating a sample thus costs N dependent forward passes. For images, x is the grid of tokens produced by a discrete tokenizer ($N = 256$ for a 16×16 grid) and the canonical order is raster (left to right, top to bottom).

Any-order generation. An any-order AR model factorizes the joint distribution for any permutation σ of the positions as

$$p_\sigma(x \mid c) \;=\; \prod_{t=1}^{N} p\big(x_{\sigma_t} \,\big|\, x_{\sigma_{<t}},\, c\big), \tag{2}$$

where σ_t is the position generated at step t and $\sigma_{<t}$ are the positions generated before it. The token at position σ_t is sampled typically by passing as input to the model a position query that attends to the tokens generated so far (Pannatier et al., 2024; Pang et al., 2025; Li et al., 2026; Zhang et al., 2026). The conditional at any position that has not yet been generated can be obtained via a single query, and many such queries can be used in one pass to decode in parallel. Given the set S of committed positions and a group of new positions $s_1, \ldots, s_m$, parallel decoding samples each token of the group independently given x_S and commits all of them at once. The group is therefore drawn from a different distribution than under sequential decoding, where each token also conditions on the earlier tokens of the group,

$$\underbrace{\prod_{j=1}^{m} p\big(x_{s_j} \,\big|\, x_S,\, c\big)}_{\text{parallel}} \;\neq\; \underbrace{\prod_{j=1}^{m} p\big(x_{s_j} \,\big|\, x_S,\, x_{s_{<j}},\, c\big)}_{\text{sequential}}. \tag{3}$$

Parallel decoding therefore samples an approximation of equation 2 whose error grows with the size of the group and with how strongly the tokens within depend on one another.

Speculative decoding. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) reduces the number of passes without changing the distribution. Given the committed prefix, a drafter q proposes tokens $\hat{x}_1, \ldots, \hat{x}_\gamma$ for the next γ positions, drawing $\hat{x}_k \sim q_k$. A single pass of the target model then evaluates, for every k, the conditional the model would have used had it generated $\hat{x}_{<k}$:

$$p_k \;=\; p\big(\,\cdot \,\big|\, x_S,\, \hat{x}_{<k},\, c\big), \tag{4}$$

which is possible in one pass because the target model is causal. A draft token k is accepted with probability $\min\big(1,\, p_k(\hat{x}_k)/q_k(\hat{x}_k)\big)$. The first rejected token is resampled from the residual distribution $\mathrm{norm}(\max(0,\, p_k - q_k))$. The resulting tokens are distributed exactly as under sequential sampling from the target p (equation 1), and the choice of q determines only the speed. Given that the earlier draft tokens were accepted, a draft k is accepted with expected probability

$$a_k \;=\; 1 - \mathrm{TV}\big(q_k,\, p_k\big), \qquad \mathrm{TV}(q, p) \;=\; \tfrac{1}{2}\sum_x \big|q(x) - p(x)\big| \;=\; 1 - \sum_x \min\big(q(x),\, p(x)\big), \tag{5}$$

where TV is the total variation between two distributions. The expected number of tokens accepted per step is then $\sum_{k=1}^{\gamma} \prod_{j \le k} a_j$.

3 MOTIVATION

Two sources of disagreement. Equation 5 shows that the disagreement between the drafter and target conditionals is quantified by their total variation distance. We denote this disagreement at draft position k by $D_k = \mathrm{TV}\big(q_k,\; p(\cdot \mid x_S, \hat{x}_{<k}, c)\big)$. A nonzero D_k can arise from either model

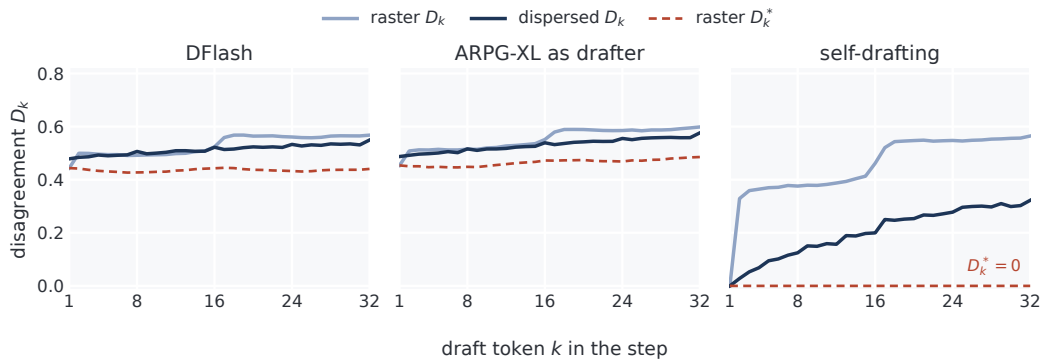

Figure 2: Disagreement with the target at each draft token of a step for three parallel drafters on ARPG-L ($\gamma = 32$), along the raster (light) and dispersed (dark) orders. D_k^*, the disagreement left when the drafter has the same conditioning as the target, is zero for self-drafting. Along raster, the jump at draft 17 is when the draft reaches the second drafted row in the 16×16 grid and thus doesn't have any adjacent committed token.

mismatch or conditioning mismatch. A sequential drafter such as EAGLE (Li et al., 2024a; Jang et al., 2025) generates each draft token conditioned on the preceding draft tokens within the same speculative step. It therefore matches the target's conditioning, and any remaining disagreement is due to model mismatch. In contrast, a parallel drafter proposes all tokens in a speculative step in a single pass, conditioning only on the committed tokens, $q_k = q(\cdot \mid x_S, c)$, so its disagreement can contain contributions from both sources. To isolate the contribution of model mismatch, we evaluate the drafter under the target's conditioning, supplying it with the preceding draft tokens $\hat{x}_{<k}$. We define

$$D_k^* \;=\; \mathrm{TV}\big(q(\cdot \mid x_S, \hat{x}_{<k}, c),\; p(\cdot \mid x_S, \hat{x}_{<k}, c)\big). \tag{6}$$

Figure 2 compares D_k and D_k^* for two parallel drafters, using the ARPG-L any-order model as the target. DFlash (Chen et al., 2026; Cheng et al., 2026) is a smaller model trained specifically for drafting, whereas ARPG-XL is a larger model from the same family as the target. Providing the target's conditioning yields only a modest reduction in average disagreement along the raster order: from 0.53 to 0.44 for DFlash and from 0.55 to 0.46 for ARPG-XL. Thus, removing the conditioning mismatch closes only a small fraction of the gap for either drafter; most of their disagreement with the target remains attributable to model mismatch.

Self-speculative decoding. When using the same model as drafter and target (self-speculative decoding or self-drafting), each draft is sampled from the target's conditional given the committed tokens only, $q_k = p(\cdot \mid x_S, c)$. Given the same conditioning as the target, the drafter's distribution is equivalent to the target's, so $D_k^* = 0$ and

$$D_k \;=\; \mathrm{TV}\Big(p\big(\cdot \mid x_S, c\big),\; p\big(\cdot \mid x_S, \hat{x}_{<k}, c\big)\Big), \tag{7}$$

which is the change in the target's conditional at position k when the earlier drafts of the step are added to the prefix. This quantity can also be interpreted as the dependence of x_k on the positions drafted before it. There are two ways to reduce it. The first is to give the drafter the missing context, which requires drafting sequentially and removes the parallelism. The second is to draft tokens that depend weakly on one another, so that the earlier drafts carry little information about x_k and the target's conditional stays close to the drafter's. Self-drafting exists for left-to-right models (Santilli et al., 2023; Teng et al., 2025; So et al., 2026), but such models can only draft the next positions, which are adjacent and strongly dependent. An any-order model can evaluate its conditional at any position that has not been generated yet, so it can draft a group of arbitrary positions from $p(\cdot \mid x_S, c)$ in one pass. Its acceptance rate is then determined by the dependence among the tokens drafted together, which starts at zero at the first draft token and grows along the step, but can be halved by the choice of order (Figure 2, right). The question is therefore how to choose positions whose tokens depend weakly on one another.

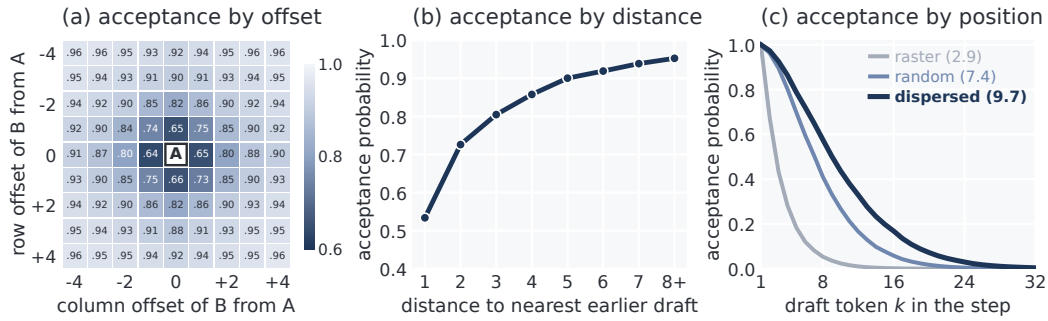

Figure 3: Dependence between drafted tokens with self-drafting (ARPG-L). (a) Acceptance probability of a draft at B after a draft at A, by grid offset. (b) Acceptance probability of a draft against its distance to the nearest earlier draft of the step ($\gamma = 32$). (c) Probability that the draft at each position of the step is accepted, along the raster, random and dispersed orders; the area under each curve (in parentheses) is the expected number of accepted tokens per step.

Dependence falls with distance. We measure how the dependence between drafted tokens varies with the distance between their positions (Figure 3). With self-drafting, a draft is accepted with probability $1 - D_k$ (equation 7), and we report this acceptance probability. First, the dependence between a pair of tokens is local. We add a single draft token at a position A to the committed context and measure the acceptance probability of a draft at every other position B. It is about 70% when B is adjacent to A and exceeds 97% when B is more than six positions away (Figure 3a). Second, the same holds in a full step of 32 drafts where the acceptance probability increases with the distance to the nearest earlier draft of the step, from $\sim 50\%$ when that draft is adjacent to above 90% at a distance of seven positions (Figure 3b). Third, consequently, the order determines how many drafts are accepted per step. With a draft window of $\gamma = 32$, the expected number of accepted tokens is 2.9 along the raster order, 7.4 along a random order, and 9.7 along a dispersed order in which consecutive positions are far apart (Figure 3c). A random order falls in between because a random set of positions can still contain close pairs. The order is significant only when self-drafting. For the other parallel drafters, the dispersed order barely lowers the disagreement: DFlash goes from 0.53 to 0.51 and ARPG-XL from 0.55 to 0.53, whereas self-drafting goes from 0.45 to 0.20 (Figure 2). These observations motivate generating along an order in which consecutive positions are as far apart as possible.

4 Method

4.1 Dispersed any-order speculative decoding

DSD is simply self-drafting with an any-order model along a dispersed order. Let σ be the generation order and S the set of positions committed so far. Each step takes the next γ positions of σ and drafts all of them in parallel given the committed tokens, $q_k = p(\cdot \mid x_S, c)$ (Figure 4d). A verification pass then evaluates the chained conditionals p_k of equation 4. The drafts are accepted in order (following the rule in Section 2) until the first rejection, where the rejected draft is resampled from the residual distribution (Algorithm 1). The committed tokens are an exact sample of p_σ in equation 2 for every order (Leviathan et al., 2023; Chen et al., 2023). The drafting and verification can be performed together so that a step costs a single target pass (Samragh et al., 2025; Liu et al., 2025).

Section 3 showed that the disagreement of self-drafting is the dependence among the positions drafted together (equation 7), and that this dependence falls with the distance between them. DSD therefore generates along an order in which the positions drafted together are far apart (Section 4.2). DSD can be combined with any self-speculative sampler (Teng et al., 2025; So et al., 2026).

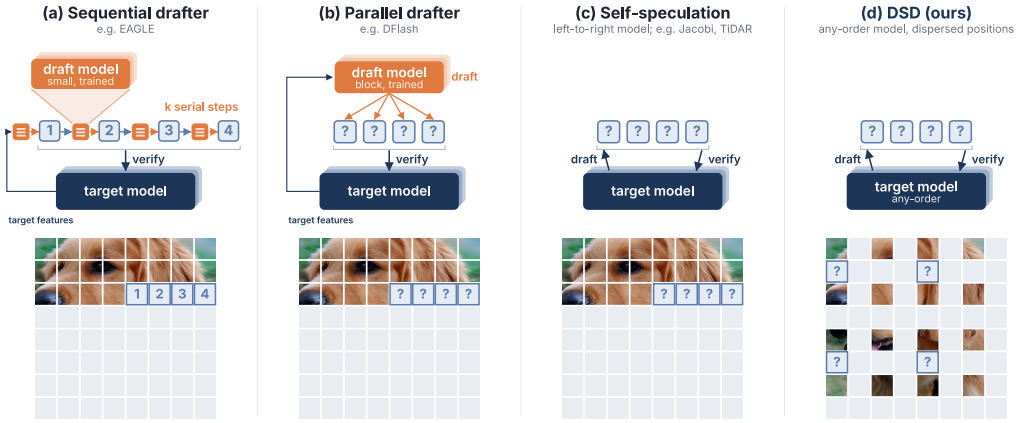

Figure 4: Speculative decoding methods. (a) A separate sequential drafter (Li et al., 2024a). (b) A separate parallel drafter (Chen et al., 2026). (c) A left-to-right model drafting its next tokens itself (Santilli et al., 2023; Liu et al., 2025). (d) DSD: an any-order model drafting dispersed positions from its own conditionals.

4.2 Choosing the order

Each step drafts the next γ positions of σ, starting where the previous step stopped. DSD therefore needs an order in which every γ consecutive positions, at every point of the generation, are spread over the positions not yet generated. We use a coarse-to-fine bit-reversal order, which generates the grid one lattice at a time and halves the spacing at each level. On a 16×16 grid, it generates a 2×2 lattice with spacing 8 first, then the rest of the 4×4 lattice with spacing 4, and so on down to spacing 1. Within each level, the new positions are ordered by the same rule, so consecutive positions are always far apart and every window of the order is spread over the image (Appendix B). The rank of a position in this order is obtained by interleaving the bits of its row and column indices (Morton, 1966) and reversing them (van der Corput, 1935).

Algorithm 1 DSD

Require: model p, condition c, order σ, window γ
$S \leftarrow \emptyset$
while $|S| < N$ **do**
  $(w_1, \ldots, w_\gamma) \leftarrow$ next γ positions of σ
  $\hat{x}_k \sim q_k = p(\cdot \mid x_S, c)$ at w_k, all k ▷ parallel
  $p_k \leftarrow p(\cdot \mid x_S, \hat{x}_{<k}, c)$ at w_k, all k ▷ parallel
  for $k = 1, \ldots, \gamma$ **do**
    if $\mathcal{U}(0,1) < p_k(\hat{x}_k)/q_k(\hat{x}_k)$ **then**
      $x_{w_k} \leftarrow \hat{x}_k$
    else
      $x_{w_k} \sim \mathrm{norm}(\max(0,\, p_k - q_k))$
      break
    end if
  end for
  $S \leftarrow S \cup \{w_1, \ldots, w_k\}$
end while
return x

5 Experiments

5.1 Setup

Models. We use seven checkpoints from three families of any-order image models, all class-conditional on ImageNet: ARPG-L and ARPG-XXL (Li et al., 2026), RandAR-L (0.3B) and RandAR-XL (0.7B) (Pang et al., 2025), and LPD-L and LPD-XXL at 256×256 resolution and LPD-L at 512×512 (Zhang et al., 2026). These are the smallest and the largest released checkpoint of each family and LPD's high-resolution model. The tokenizer maps a 256×256 image to 256 tokens and a 512×512 image to $1{,}024$. Every model uses its released optimal classifier-free guidance (CFG) scale.

Samplers. DSD is compatible with various self-drafting methods. Unless stated otherwise we run it with SCD (So et al., 2026), which needs the fewest passes of the samplers in Table 2, and with a draft window of $\gamma = 64$ (128 for the $1{,}024$-token model). We compare DSD with sequential

Table 1: Target passes per image of exact samplers. The trained drafters and the self-speculative samplers run along the raster order and DSD along the dispersed order. $\gamma = 128$ is used for SJD, SCD and DSD on LPD-L 512, $\gamma = 32$ for the trained drafters, and $\gamma = 64$ for the rest. The trained drafter passes are not counted.

		trained drafter		self-speculative				reduction by DSD	
model	sequential	EAGLE	DFlash	SJD	fused	SCD	**DSD**	vs. sequential	vs. best other
ARPG-L	256	133.3	115.6	117.7	90.2	71.4	**21.6**	11.9×	3.3×
ARPG-XXL	256	–	–	136.0	109.7	95.0	**26.9**	9.5×	3.5×
RandAR-0.3B	256	–	–	125.7	98.2	80.1	**22.7**	11.3×	3.5×
RandAR-0.7B	256	169.2	140.9	133.1	108.1	89.4	**25.4**	10.1×	3.5×
LPD-L 256	256	–	–	111.2	86.6	63.5	**20.9**	12.2×	3.0×
LPD-XXL 256	256	–	–	115.9	89.8	69.4	**22.6**	11.3×	3.1×
LPD-L 512	1024	–	–	420.2	321.5	159.3	**34.4**	29.8×	4.6×

Table 2: Target passes per image of each self-speculative sampler along the dispersed order, and in parentheses the reduction relative to the same sampler along raster.

sampler	ARPG-L	ARPG-XXL	RandAR-0.3B	RandAR-0.7B	LPD-L 256	LPD-XXL 256	LPD-L 512
SJD	30.6 (3.8×)	36.0 (3.8×)	31.5 (4.0×)	34.8 (3.8×)	29.1 (3.8×)	31.0 (3.7×)	57.1 (7.4×)
fused	28.7 (3.1×)	35.5 (3.1×)	28.9 (3.4×)	32.8 (3.3×)	26.8 (3.2×)	29.8 (3.0×)	49.5 (6.5×)
SCD	21.6 (3.3×)	26.9 (3.5×)	22.7 (3.5×)	25.4 (3.5×)	20.9 (3.0×)	22.6 (3.1×)	34.4 (4.6×)

decoding, with the self-speculative samplers SJD (Teng et al., 2025), SCD and fused drafting, also known as quadratic decoding (Samragh et al., 2025; Liu et al., 2025) along the raster order (as published), with the trained drafters DFlash (Chen et al., 2026) and EAGLE (Li et al., 2024a), and with the lossy sampling.

Metrics. We measure cost as the number of target forward passes per image, which does not depend on the hardware, and report wall-clock time in Appendix D. For the trained drafters, only the target's passes are counted. We measure sample quality on $50{,}000$ images with FID and with FDr^6 (Yang et al., 2026), which averages normalized Fréchet distances over six image representations.

5.2 Comparison with exact samplers

Main results. Table 1 compares DSD with every exact sampler. DSD needs the fewest target passes on every model. On the 256-token models it samples the model's distribution in 21 to 27 passes, 9.5 to 12.2× fewer than sequential decoding and 3.0 to 3.5× fewer than the best published sampler, SCD along the raster order. The trained drafters (Appendix A) accept about two tokens per step and need more passes than every self-speculative sampler. DSD needs 5.4× fewer passes than the best trained drafter on ARPG-L and 5.5× fewer on RandAR-0.7B. Changing the order from raster to dispersed barely changes the acceptance for trained drafters, whereas it improves self-drafting by 2.6× (Figure 5b). The gain grows with the number of tokens: on LPD-L at 512×512 resolution, which generates $1{,}024$ tokens, DSD needs 34 passes, 30× fewer than sequential decoding and 4.6× fewer than SCD.

The order and the sampler. Table 2 isolates the order. It runs each self-speculative sampler along the dispersed order, with nothing else changed. Every sampler needs 3.0 to 4.0× fewer passes than along raster on the 256-token models, and 4.6 to 7.4× fewer on the $1{,}024$-token model. SCD needs the fewest passes along both orders, and DSD uses it by default.

Draft window. Figure 5(a) varies the draft window γ on ARPG-L. Along the raster order SCD reduces the number of passes by 2.5× at $\gamma = 8$ and by 3.4× at $\gamma = 32$, and it stays at 3.6× from $\gamma = 64$ on, because a window of adjacent positions is rejected within its first few drafts however many are proposed. Along the dispersed order the drafts late in the window are still accepted, and

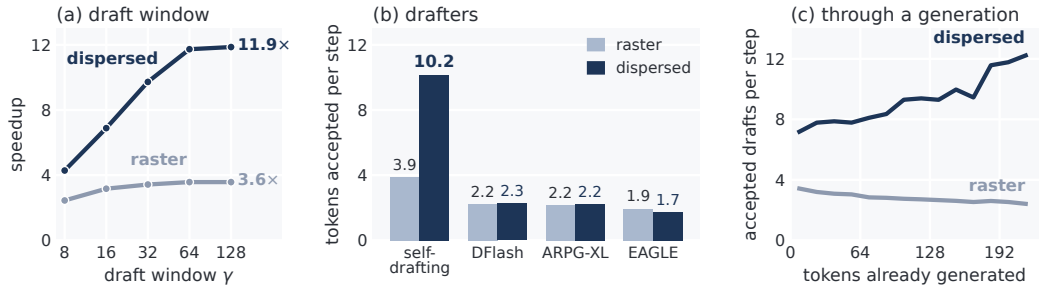

Figure 5: ARPG-L. (a) Speedup in target forward passes over sequential decoding against the draft window γ, for SCD. (b) Tokens accepted per step at $\gamma = 32$ for self-drafting and three separate drafters, along the raster (light) and dispersed (dark) orders. (c) Accepted drafts per step with self-drafting as the image fills in ($\gamma = 32$).

the reduction grows from $4.3\times$ at $\gamma = 8$ to $9.7\times$ at $\gamma = 32$ and $11.7\times$ at $\gamma = 64$. This behavior is present in all tested models.

Acceptance dynamics. Along the dispersed order, a step accepts about 7 drafts at the start of an image and about 12 near the end (Figure 5c), since each draft has more committed neighbours as the image fills in and depends less on the other drafts of the step. Along the raster order, the most informative neighbour of a draft is the draft to its left, and a step accepts about 3 drafts throughout.

5.3 Comparison with lossy sampling

In order to speed up inference, each any-order model can be sampled in a lossy manner by generating multiple tokens in parallel (equation 3). It approaches the sequential distribution as the number of steps grows. DSD, on the other hand, speeds up inference while sampling from the exact sequential distribution. Figure 6 places the two on the same axes for RandAR-0.7B, and Figure 9 for the largest checkpoint of each family.

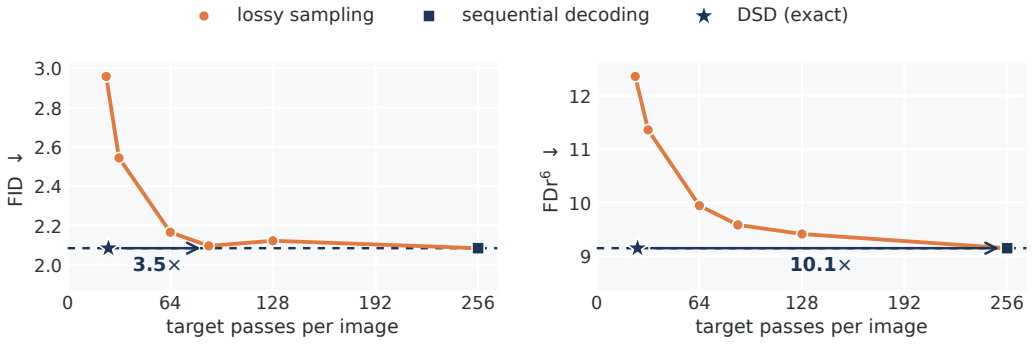

Figure 6: FID (left) and FDr^6 (right) against target passes on RandAR-0.7B. The dashed line shows the model's distribution (which DSD matches). Arrows go from DSD to the first lossy point within noise of it.

On FID (Figure 6, left), lossy sampling reaches DSD's value at 88 steps on RandAR-0.7B, at 128 steps on ARPG-XXL and at 32 steps on LPD-XXL, which is 3.5, 4.1 and $1.4\times$ DSD's passes. FDr^6 averages the Fréchet distances of six image representations and is a more reliable proxy than FID, which relies on a single Inception representation (Yang et al., 2026). On FDr^6 (Figure 6, right), lossy sampling reaches DSD's value only at sequential decoding on RandAR-0.7B and ARPG-XXL, 10.1 and $8.3\times$ DSD's passes, and at 128 steps on LPD-XXL, $5.2\times$. Where lossy sampling ties DSD

on FID (Inception representation), the other representations still favor DSD on all three models (Appendix C). At the step counts where lossy sampling matches DSD on FID, DSD is 3.2, 3.3 and $1.5\times$ faster in wall-clock time (Appendix D).

FID and FDr^6 are proxies for sample quality, and lossy sampling can match DSD on them while sampling a different distribution. The distribution the model defines, which sequential decoding samples, is usually the one preferred, since it does not accumulate the errors of sampling nearby tokens independently. DSD samples it exactly, whereas lossy sampling remains away from it even where the metrics match. The model's own likelihood measures this directly: the KL divergence between the distribution of lossy sampling and the model's (sequential) shrinks as the number of steps grows but remains well above zero at every step count we ran, whereas DSD's samples cannot be told apart from those of sequential decoding (Appendix C).

6 Related work

Speculative decoding. Speculative decoding keeps the target's distribution exact by verifying the drafts against it (Leviathan et al., 2023; Chen et al., 2023). The drafts can come from a separate model or from heads trained to predict the target's next tokens, sequentially (Li et al., 2024a;b; 2025) or in parallel (Cai et al., 2024; Chen et al., 2026; Cheng et al., 2026). On image models, several methods relax the verification to accept more drafts, which gives up exactness (Jang et al., 2025; Park et al., 2025; So et al., 2025; He et al., 2025). Self-speculative decoding drafts with the target itself, either by refining a window of drafts across steps with Jacobi iterations (Santilli et al., 2023; Teng et al., 2025; So et al., 2026; Kang et al., 2026) or by training the model to both draft and verify (Samragh et al., 2025; Liu et al., 2025). These methods were developed for left-to-right models, whose drafts are the next contiguous positions.

Any-order speculative decoding. Two methods apply speculative decoding to models that are not left-to-right, both for text. Neither has been explored for images, where any-order models retain good sample quality and are widely used. σ-GPT (Pannatier et al., 2024) drafts from an any-order model's own conditionals and selects, among several candidate orders, the one with the most accepted drafts, but this selection biases the output, so its samples are not exact. Any-subset speculative decoding (Guo & Ermon, 2025) is exact and applies speculative decoding to an any-subset model, which can generate any positions but must draft them in ascending order. DSD uses an any-order model, which lets it choose the order in which positions are drafted.

Dispersed generation orders. Some lossy parallel decoders for images order the positions so that the tokens generated in one step are far apart (Zhang et al., 2026; Besnier et al., 2025), and dilated unmasking schedules do the same for masked diffusion language models (Luxembourg et al., 2026). These methods sample the tokens of a step independently and do not verify them, so their output only approximates the model's distribution. DSD uses a dispersed order to make an any-order model's drafts agree with its own verification, and its output remains an exact sample. DSD works with any order that keeps the positions drafted in every step far apart (Section 4.2).

7 Conclusion

The disagreement between a drafter and the target has two sources, the model and the conditioning, and self-drafting removes the first. For an any-order model, the disagreement that remains is the dependence among the positions drafted together, which falls with the distance between them. DSD drafts along a dispersed order and samples the model's exact distribution with up to $30\times$ fewer target passes than sequential decoding, without training. On FDr^6, the models' lossy parallel sampling needs 5.2–$10.1\times$ more steps to reach its quality.

Limitations. DSD requires an any-order model, since a left-to-right model cannot evaluate its conditionals at arbitrary positions. Its samples are exact with respect to the model's distribution along the order used. We measure wall-clock time at batch size 1, the standard setting for speculative decoding. At large batch sizes the forward pass becomes compute-bound, and the speedup of speculative decoding naturally shrinks.

AI USE STATEMENT

In this work, we used generative AI tools to aid writing and assist with finding related work. We have not used generative AI tools for research ideation or to help develop theoretical models or conceptual frameworks.

REFERENCES

Victor Besnier, Mickael Chen, David Hurych, Eduardo Valle, and Matthieu Cord. Halton scheduler for masked generative image transformer. In *International Conference on Learning Representations (ICLR)*, 2025.

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 (ICML)*, 2024.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. MaskGIT: Masked generative image transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

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.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *International Conference on Machine Learning (ICML)*, 2026.

Xin Cheng, Xingkai Yu, Chenze Shao, Jiashi Li, Yunfan Xiong, et al. DSpark: Confidence-scheduled speculative decoding with semi-autoregressive generation. *arXiv preprint arXiv:2607.05147*, 2026.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *International Conference on Machine Learning (ICML)*, 2024.

Gabe Guo and Stefano Ermon. Reviving any-subset autoregressive models with principled parallel sampling and speculative decoding. *arXiv preprint arXiv:2504.20456*, 2025.

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 (ICML)*, 2025.

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 *International Conference on Learning Representations (ICLR)*, 2025.

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

Tanishq Kumar, Tri Dao, and Avner May. Speculative speculative decoding. In *International Conference on Learning Representations (ICLR)*, 2026.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning (ICML)*, 2023.

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

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning (ICML)*, 2024a.

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 (EMNLP)*, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Jingyu Liu, Xin Dong, Zhifan Ye, Rishabh Mehta, Yonggan Fu, Vartika Singh, Jan Kautz, Ce Zhang, and Pavlo Molchanov. TiDAR: Think in diffusion, talk in autoregression. *arXiv preprint arXiv:2511.08923*, 2025.

Omer Luxembourg, Haim Permuter, and Eliya Nachmani. Plan for speed: Dilated scheduling for masked diffusion language models. In *International Conference on Machine Learning (ICML)*, 2026.

Guy M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. Technical report, IBM Ltd., Ottawa, Canada, 1966.

Ziqi Pang, Tianyuan Zhang, Fujun Luan, Yunze Man, Hao Tan, Kai Zhang, William T Freeman, and Yu-Xiong Wang. RandAR: Decoder-only autoregressive visual generation in random orders. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Arnaud Pannatier, Evann Courdier, and François Fleuret. σ-GPTs: A new approach to autoregressive models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, 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. In *ICLR Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models (SCOPE)*, 2025.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In *Proceedings of Machine Learning and Systems (MLSys)*, 2023.

Mohammad Samragh, Arnav Kundu, David Harrison, Kumari Nishu, Devang Naik, Minsik Cho, and Mehrdad Farajtabar. Your LLM knows the future: Uncovering its multi-token prediction potential. *arXiv preprint arXiv:2507.11851*, 2025.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodolà. Accelerating transformer inference for translation via parallel decoding. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

Junhyuk So, Juncheol Shin, Hyunho Kook, and Eunhyeok Park. Grouped speculative decoding for autoregressive image generation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

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

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. SpecTr: Fast speculative decoding via optimal transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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 (ICLR)*, 2025.

Johannes G. van der Corput. Verteilungsfunktionen (Erste Mitteilung). *Proceedings of the Koninklijke Akademie van Wetenschappen te Amsterdam*, 38:813–821, 1935.

Jiawei Yang, Zhengyang Geng, Xuan Ju, Yonglong Tian, and Yue Wang. Representation Fréchet loss for visual generation. *arXiv preprint arXiv:2604.28190*, 2026.

Ming Yin, Minshuo Chen, Kaixuan Huang, and Mengdi Wang. A theoretical perspective for speculative decoding algorithm. In *Advances in Neural Information Processing Systems (NeurIPS)*, 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. In *International Conference on Learning Representations (ICLR)*, 2026.

A Experimental setup

Models. We use the released checkpoints of Table 3, trained as described in their papers. All seven use the LlamaGen VQ tokenizer (Sun et al., 2024).

Table 3: Any-order model checkpoints.

model	parameters	tokens
ARPG-L (Li et al., 2026)	320M	16×16
ARPG-XXL (Li et al., 2026)	1.3B	16×16
RandAR-L (Pang et al., 2025)	0.3B	16×16
RandAR-XL (Pang et al., 2025)	0.7B	16×16
LPD-L (Zhang et al., 2026)	337M	16×16
LPD-XXL (Zhang et al., 2026)	1.4B	16×16
LPD-L (Zhang et al., 2026)	337M	32×32

Trained drafters. We train EAGLE (Li et al., 2024a) and DFlash (Chen et al., 2026) drafters for ARPG-L and RandAR-0.7B. Both get as input the target's hidden states at the committed positions and use the target's frozen output head. The EAGLE drafter is a single transformer layer that drafts one position at a time from the target's last hidden state and its own previous state. The position of the next draft is given by the target's positional encoding. On ARPG-L it drafts with LANTERN++'s static tree (Park et al., 2025) (15M parameters). The DFlash drafter drafts all positions of a step in one pass with transformer layers at the target's width (4 layers and 57M parameters for ARPG-L, 2 layers for RandAR-0.7B). The hidden states of five target layers are injected into the layers as keys and values. Drafters are trained on ImageNet with a KL loss to match the target's distribution at each drafted position, the EAGLE drafter given the committed tokens and the earlier drafts of the step, and the DFlash drafter given the committed tokens only, since it drafts all positions at once. We use AdamW with learning rate 10^{-4} and train until the loss plateaus. ARPG-XL, a larger released model of the same family (719M parameters), is used as a drafter as is.

Self-speculative samplers. Each step costs one pass of the target as well as the cost of drafting, but the two can be parallelized. With a separate drafter, the likely outcomes of the current verification can be predicted and drafted in advance on separate hardware (Kumar et al., 2026). With self-speculative decoding (same model drafting and verifying), the next drafts can be produced inside the verification pass either for every possible acceptance outcome of the current step (Samragh et al., 2025; Liu et al., 2025), or, in the Jacobi family, from previous drafts that are adjusted from step to step (Santilli et al., 2023; Teng et al., 2025; So et al., 2026). In every case a step then costs a single target pass. The experiments of Section 5 use samplers of the second kind: fused drafting, also known as quadratic decoding (Samragh et al., 2025; Liu et al., 2025), SJD (Teng et al., 2025) and SCD (So et al., 2026).

Measuring the disagreement. For Figure 2 we run the drafting loop on 64 images at $\gamma = 32$ and record, at every draft, D_k from the drafter's distribution and the target's chained conditional. D_k^*

restarts the drafter from the extended prefix $(x_S, \hat{x}_{<k})$ and takes its distribution at position k. The tokens per step are computed by $\sum_k \prod_{j \le k}(1 - D_j)$ and reproduce the measured ones within 2% for every drafter.

Hardware. The results of Section 5 were obtained on an NVIDIA H100 GPU (80 GB).

B THE BIT-REVERSAL ORDER

Construction. The order fills the grid from coarse to fine. It starts at the top-left corner and then proceeds in levels, each of which fills the gaps between the positions placed so far. At each level, the spacing is halved, starting from half the grid width, and the order goes through the positions placed so far three times, in the order they were placed: it places a new position at the new spacing below each of them, then one to the right of each, then one diagonally from each. Figure 7 shows the three levels of an 8×8 grid. The first level places ranks 1–3 at a distance of 4 below, right of and diagonally from rank 0. The second level places ranks 4–7 two rows below ranks 0–3, ranks 8–11 two columns to their right and ranks 12–15 diagonally, and the third level does the same from ranks 0–15 at a distance of 1. Since each pass goes through positions that are already spread over the grid, consecutive positions in the order are always at least half the grid width apart. For a $G \times G$ grid with G a power of two, the same order is obtained in closed form by interleaving the binary digits of the row and column of each position (Morton, 1966), reversing them (van der Corput, 1935), and sorting the positions by the result.



Figure 7: The bit-reversal order on an 8×8 grid, one level per panel. Each level shifts the positions placed so far (grey) down, right and diagonally by half the previous spacing, keeping their order. Numbers are ranks in the order, and arrows go from rank 0 to its three shifted copies.

Acceptance. Figure 8 shows the drafts of four steps of one generation along the raster and bit-reversal orders. Along raster the drafts of a step are adjacent and have low acceptance probabilities, whereas along the bit-reversal order they are spread over the image and have high ones.

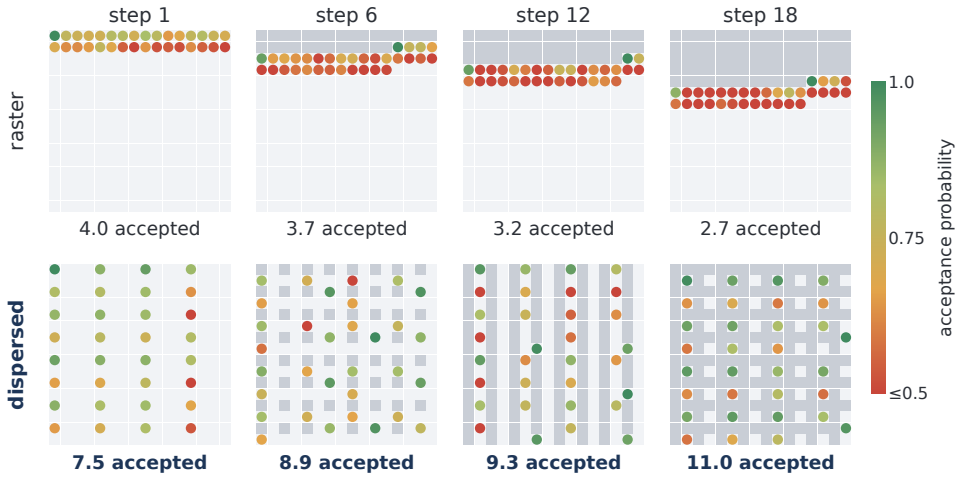

Figure 8: Steps 1, 6, 12 and 18 of one generation on ARPG-L along the raster (top) and bit-reversal (bottom) orders, $\gamma = 32$. Committed tokens are grey and the step's drafts are colored by their acceptance probability $1 - \mathrm{TV}(q_k, p_k)$; the number under each grid is the expected number of accepted drafts.

C Comparison with lossy sampling

This section expands the comparison of Section 5.3. Figure 9 gives the comparison of Figure 6 for the largest checkpoint of each family.

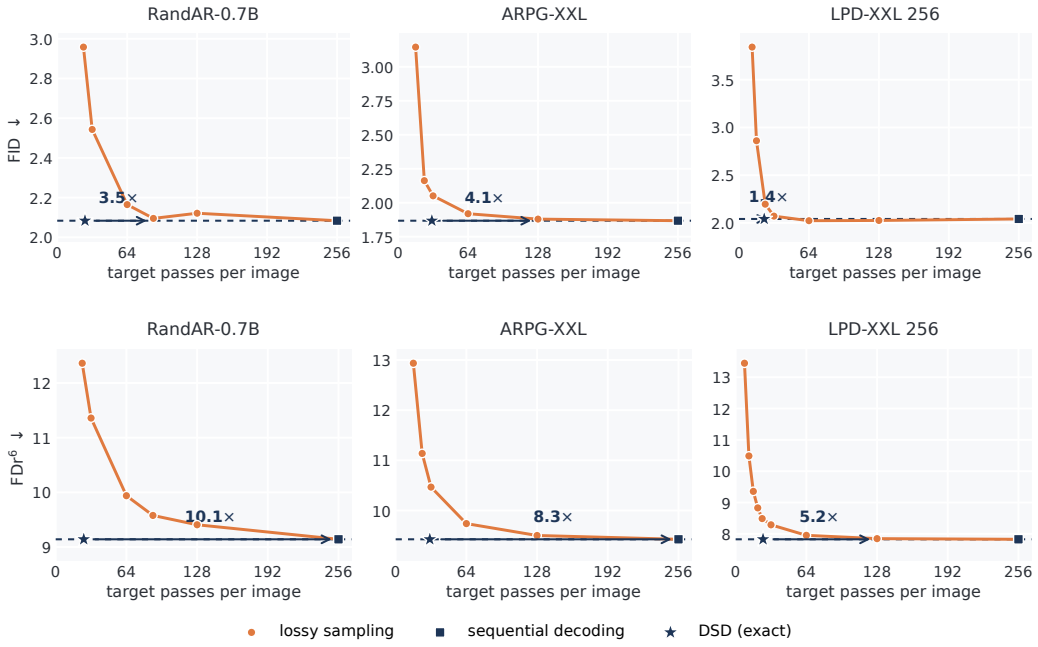

Figure 9: FID (top) and FDr^6 (bottom) against target passes for the largest checkpoint of each family. Orange: the model's lossy sampler; its last point (square) is sequential decoding. Star and dashed line: DSD and the model's distribution. Arrows go from DSD to the first lossy point within noise of it.

The components of FDr^6. Table 4 splits FDr^6 into its six representations at the step counts where the FID of lossy sampling first ties DSD's (Figure 9, top). There the Inception and ConvNeXt components differ by at most 0.12, while the MAE, DINOv2, SigLIP and CLIP components still place the samples of lossy sampling further from ImageNet, by 0.19 to 1.50, on all three models.

Table 4: The six components of FDr^6 at the step count where the FID of lossy sampling first ties DSD's (Figure 9, top). The table shows the normalized Fréchet distance of lossy sampling minus DSD's, so positive values favor DSD. The last column is their mean.

model (steps)	Inception	ConvNeXt	MAE	DINOv2	SigLIP	CLIP	FDr^6
RandAR-0.7B (88)	$+0.01$	$+0.05$	$+0.36$	$+0.51$	$+0.86$	$+0.83$	$+0.44$
ARPG-XXL (128)	$+0.01$	$+0.02$	$+0.19$	$+0.21$	$+0.48$	$+0.39$	$+0.22$
LPD-XXL 256 (32)	$+0.02$	$+0.12$	$+0.68$	$+0.91$	$+1.50$	$+1.38$	$+0.77$

Distance from the model's distribution. FID and FDr^6 are not the most accurate measure of how far lossy sampling is from the sequential distribution of the model. The model's own probabilities measure this directly. For an image x, a single teacher-forced pass gives both $\log p_\sigma(x)$, its log-probability under sequential sampling along the order σ, and $\log q(x)$, its log-probability under lossy sampling along the same order. The KL divergence of lossy sampling from the model's distribution,

$$\mathrm{KL}(q \,\|\, p_\sigma) \;=\; \mathbb{E}_{x\sim q}\big[\log q(x) - \log p_\sigma(x)\big], \tag{8}$$

is estimated by averaging $\log q(x) - \log p_\sigma(x)$ over samples of lossy sampling. Using the same log-ratio as a score, we also report the AUC for telling samples of lossy sampling from samples of DSD (0.5 when they cannot be told apart). Table 5 gives both at 64 steps, with lossy sampling along the same order and guidance as DSD, on $2{,}500$ images per sampler. Lossy sampling is 2.0 to 8.9 nats per image from the model's distribution, and its samples are told apart from those of DSD with an AUC of 0.90 to 0.996. DSD and sequential decoding, which sample the same distribution, give an AUC of 0.48 to 0.52.

Table 5: KL divergence between lossy sampling (64 steps) and the model's distribution, and AUC for telling lossy samples from those of DSD. The last column is the same AUC between sequential decoding and DSD.

		AUC	
model	KL (nats per image)	lossy vs. DSD	sequential vs. DSD
ARPG-L	5.3	0.984	0.48
ARPG-XL	6.4	0.989	0.52
RandAR-0.3B	2.0	0.904	0.51
LPD-L 256	8.9	0.996	0.50

D LATENCY

Following the standard evaluation of speculative decoding (Leviathan et al., 2023; Li et al., 2024a), we measure the time per image at batch size 1, where each pass is limited by memory bandwidth. Table 6 compares DSD with sequential decoding, which samples the same distribution, on all seven models: DSD is 7.6 to $27\times$ faster. Table 7 compares it with the lossy samplers of Figure 6 at the fewest steps at which their FID matches DSD's: DSD is 3.2, 3.3 and $1.5\times$ faster on RandAR-0.7B, ARPG-XXL and LPD-XXL, although lossy samples do not follow the model's sequential distribution.

Table 6: Seconds per image at batch size 1 on one H100, for the two exact samplers.

model	sequential	DSD	speedup
RandAR-0.3B	4.19	**0.40**	10.6×
RandAR-0.7B	6.26	**0.67**	9.4×
ARPG-L	3.42	**0.36**	9.6×
ARPG-XXL	6.55	**0.86**	7.6×
LPD-L 256	2.33	**0.17**	13.7×
LPD-XXL 256	4.58	**0.37**	12.2×
LPD-L 512	9.26	**0.34**	27.0×

Table 7: Seconds per image at batch size 1 on an H100. Lossy: the lossy sampler of Figure 6 at the fewest steps (in parentheses) at which its FID matches DSD's.

	exact			DSD speedup	
model	sequential	DSD	lossy (steps)	vs. sequential	vs. lossy
RandAR-0.7B	6.26	**0.67**	2.16 (88)	**9.4**×	3.2×
ARPG-XXL	6.55	**0.86**	2.83 (128)	**7.6**×	3.3×
LPD-XXL 256	4.58	**0.37**	0.57 (32)	**12.2**×	1.5×