DISTILLING SPECIALIZED ORDERS FOR VISUAL GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive (AR) image generators inherit the scalability and mature inference stack of large language models, yet most of them commit to a fixed raster-scan order. This choice precludes zero-shot inpainting, outpainting, and editing without retraining. Any-order AR models remove this restriction by learning to generate under arbitrary token permutations, but they must spread a finite capacity over $N!$ factorizations and consequently trail fixed-order models in sample quality. We introduce *Ordered Autoregressive* (OAR) generation, a self-distillation procedure that resolves this tension. Starting from a pretrained any-order model, OAR extracts a specialized, content-dependent generation order for every training image from the model's own confidence scores, and then fine-tunes the model on these image–order pairs. Order specialization reallocates capacity from the full permutation space to a single high-confidence trajectory per image, while leaving the any-order conditionals, and hence the model's flexibility, intact. On class-conditional ImageNet 256×256, OAR improves FID from 2.39 to 2.17 over its any-order baseline under an identical evaluation protocol, and outperforms a random-to-raster annealing schedule applied to the same backbone (2.34). The same model performs zero-shot inpainting and outpainting and surpasses the bidirectional MaskGIT and MAR models on both tasks. It also retains multi-token parallel decoding. OAR generalizes to text-to-image generation with a different, scratch-trained backbone (FID $4.58 \rightarrow 2.56$ on Fashion Products and $1.94 \rightarrow 1.41$ on Multimodal CelebA-HQ), and human raters prefer its samples over the baseline in 64% of comparisons. On ImageNet, the procedure adds a fine-tuning stage of at most $\sim 5\%$ of the pretraining epochs, and it requires no architectural change or extra annotation.

1 INTRODUCTION

Autoregressive (AR) transformers are the dominant paradigm for language modeling (OpenAI, 2020) and underpin modern large language models (OpenAI., 2024; Team & Google., 2024; Team & Group., 2024; Yang et al., 2025). A mature systems ecosystem, including KV caching, pipeline and tensor parallelism, and efficient attention kernels, makes them highly scalable in practice (Anthropic, 2024; Dao, 2024; Kwon et al., 2023). This success has motivated AR models for image synthesis (Ramesh et al., 2021; Esser et al., 2021; Sun et al., 2024). However, most AR image generators commit to a fixed raster-scan order. Because every conditional is learned with respect to this single order, such models cannot natively perform inpainting, outpainting, or region-based editing without task-specific retraining. The generation order acts as a rigid inductive bias rather than a degree of freedom that could be steered.

AR models factorize the joint distribution of an image into a chain of conditionals, which is more tractable to learn than the joint itself (Chen et al., 2018; Shih et al., 2022). Standard models commit to one permutation of the image patches. Training over *many* permutations instead yields an any-order model that can predict any patch from any subset of the others, so inpainting, outpainting, and editing reduce to different choices of conditioning set (Pang et al., 2025; Pannatier et al., 2024). This flexibility has a cost. In principle, every factorization recovers the same joint distribution. In practice, the model must fit conditionals for up to $N!$ orderings with finite capacity, and different orderings of the same data lead to markedly different learned models (Vinyals et al., 2016). As a result, any-order models typically trail fixed-order models in sample quality. This raises the question that motivates our

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

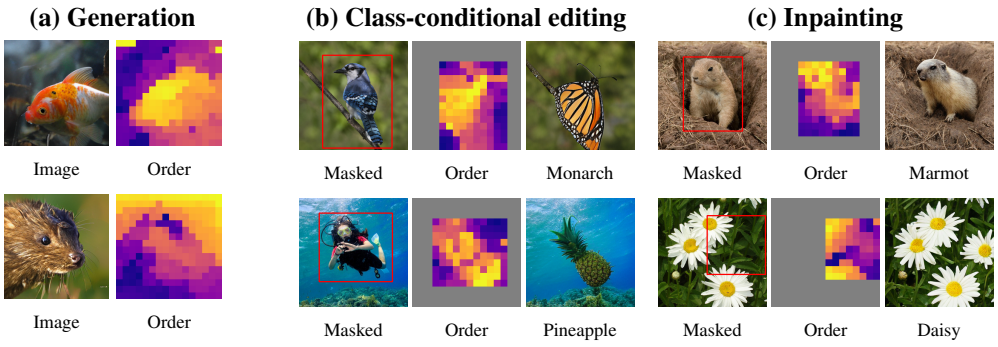

Figure 1: **A single OAR model generates, edits, and inpaints.** Orders are shown as a gradient from yellow (early) to violet (late). **(a)** Class-conditional samples with their specialized orders. **(b)** Zero-shot editing: masked regions are regenerated under a new class label, and the order adapts to the new content. **(c)** Zero-shot inpainting under the original label, yielding either a coherent reconstruction (marmot) or a novel completion (daisy). None of these tasks is seen during training.

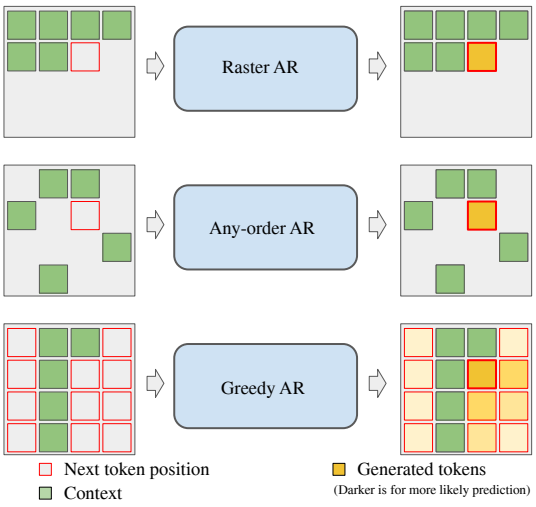

Figure 2: **Three AR generation schemes. (Top)** Raster AR generates tokens from top-left to bottom-right; each input carries content x_i and position l_i. **(Middle)** Any-order AR can generate a token at any location; the next position is supplied as an extra token or positional embedding. **(Bottom)** Greedy order selection queries the any-order model at every free position and commits only the most confident one (dark yellow).

work: *can a single AR model, trained to generate in any order, retain this flexibility while discovering specialized orders that improve generation quality?*

We show that it can. We present *Ordered Autoregressive* (OAR) generation, a three-stage self-distillation procedure built on recent any-order AR models (Li et al., 2024; Pannatier et al., 2024; Pang et al., 2025). First, an AR model is trained on random permutations to obtain an any-order model. Second, the model *relabels* its own training set: for every image, it greedily selects at each step the position that it predicts with the highest confidence, which yields a specialized, content-dependent order (Figure 2, bottom). Third, the model is fine-tuned on these image–order pairs. Fine-tuning redirects capacity from the full permutation space to one high-confidence trajectory per image. A reduced learning rate and an order-annealing schedule preserve the any-order conditionals. The resulting model generates higher-quality images along its specialized orders (Figure 1a) and still performs editing and inpainting zero-shot (Figure 1b,c).

In summary, our contributions are as follows.

- We propose OAR, a self-distillation procedure that extracts specialized generation orders from an any-order AR model's own confidence scores and fine-tunes the model on them. This improves generation quality while preserving zero-shot inpainting, outpainting, editing, and parallel decoding.
- We interpret order specialization as *capacity reallocation*: redirecting capacity from $N!$ permutations to one specialized trajectory per image improves quality without erasing the any-order conditionals. We support this view with controlled experiments that isolate the choice of order from the training schedule, including a RAR-style random-to-raster annealing baseline on the same backbone.
- On ImageNet 256×256, OAR improves FID over its any-order baseline from 2.39 to 2.17. It outperforms the bidirectional MaskGIT and MAR models on zero-shot inpainting and outpainting, and human raters prefer its samples in 64% of comparisons. On Fashion Products and Multimodal CelebA-HQ, the same procedure transfers to text-conditioned generation with a different, scratch-trained backbone.

2 Related Work

Autoregressive image generation. Early AR image models generate pixels sequentially with RNNs (Van Den Oord et al., 2016), CNNs (Van den Oord et al., 2016; Chen et al., 2018), and later transformers (Parmar et al., 2018; Chen et al., 2020). Tokenizing images into discrete patch codes with learned codebooks brought substantial gains (Van Den Oord et al., 2017; Razavi et al., 2019; Esser et al., 2021; Ramesh et al., 2021). Recent AR models rival diffusion models in sample quality (Yu et al., 2024a; Sun et al., 2024; Tian et al., 2024). Work in both language and vision nevertheless shows that the sequence order strongly affects what an AR model learns (Vinyals et al., 2016; Ford et al., 2018; Papadopoulos et al., 2024). This motivates treating the order as an object of study rather than a fixed convention.

Any-order and masked generation. Any-order AR models (Uria et al., 2016; Yang et al., 2019; Hoogeboom et al., 2022) support generation under arbitrary token orders. In vision, MaskGIT (Chang et al., 2022) and MAGE (Li et al., 2023) adopt BERT-style masked modeling (Devlin et al., 2019) with confidence-driven iterative decoding. MAR (Li et al., 2024; Fan et al., 2024) interprets masked generation as next-token prediction under flexible orders. These methods rely on *bidirectional* encoders. The set of visible tokens changes at every step, so they cannot reuse keys and values through standard KV caching. OAR instead is a *causal decoder*, which makes it directly compatible with the inference stack of modern LLMs (KV caching, tensor and token parallelism). Among causal decoders, σ-GPT (Pannatier et al., 2024) and RandAR (Pang et al., 2025) condition each prediction on the position of the next token and thereby learn any-order generation. We build on the publicly released RandAR checkpoints. RAR (Yu et al., 2024b) trains on random orders annealed toward raster order and uses the permutations mainly as a regularizer. Its final model is tied to a single fixed order. It also adds orthogonal modifications, such as scheduled classifier-free guidance and AdaLN-Zero conditioning, that affect quality independently of the order. OAR, by contrast, anneals toward *specialized, content-dependent* orders and aims to preserve any-order flexibility.

Learning generation orders. Several works in language modeling move beyond left-to-right decoding through bidirectional (Sun et al., 2017; Mehri & Sigal, 2018), syntax-tree-based (Yamada & Knight, 2001; Wang et al., 2018), or non-monotonic (Welleck et al., 2019) generation. These approaches, however, rely on hand-designed heuristics rather than orders specialized to each sample. InDIGO (Gu et al., 2019) uses insertion operations to generate in arbitrary order. Variational Order Inference (Li et al., 2021) and LO-ARM (Wang et al., 2025) learn per-sample orders by variational inference. This requires joint training of an order encoder and a generator, together with high-variance gradient estimators such as REINFORCE, because order sampling is discrete. OAR decouples order discovery from generation. It greedily extracts orders from a trained any-order model and distills them back into the same model, which avoids joint optimization and high-variance estimators.

3 WHY GENERATION ORDER MATTERS

3.1 ANY-ORDER AUTOREGRESSIVE MODELING

Let $\mathbf{x} = [x_1, \ldots, x_N]$ be the sequence of discrete tokens that represents an image. An AR model factorizes the joint distribution via the chain rule,

$$p_\theta(\mathbf{x} \mid \mathbf{l}) = \prod_{i=1}^{N} p_\theta\big(x_{l_i} \mid \mathbf{x}_{\mathbf{l}_{<i}}\big), \tag{1}$$

where $\mathbf{l}$ is a permutation of $\{1, \ldots, N\}$, and l_i and $\mathbf{l}_{<i}$ denote the i-th position and the first $i-1$ positions of the ordering. Samples are drawn token by token along $\mathbf{l}$. If the model captured the true joint exactly, the choice of $\mathbf{l}$ would be irrelevant. For learned models, however, a single fixed order such as raster scan imposes an inductive bias that may not match the structure of the data, especially for data without a natural sequential structure (Vinyals et al., 2016).

Any-order AR models instead train on orderings drawn uniformly from the set $\mathcal{P}_N$ of all $N!$ permutations and maximize $\mathbb{E}_{\mathbf{l}\sim\mathcal{U}(\mathcal{P}_N)}\big[\log p_\theta(\mathbf{x} \mid \mathbf{l})\big]$. Treating $\mathbf{l}$ as a latent variable, this objective is a lower bound on the marginal log-likelihood (Uria et al., 2014):

$$\log p_\theta(\mathbf{x}) = \log \sum_{\mathbf{l}\in\mathcal{P}_N} p(\mathbf{l})\, p_\theta(\mathbf{x} \mid \mathbf{l}) \;\geq\; \mathbb{E}_{p(\mathbf{l})}\big[\log p_\theta(\mathbf{x} \mid \mathbf{l})\big]. \tag{2}$$

In practice, the expectation is estimated by sampling one permutation per training example.

3.2 ORDER AS A PATH THROUGH A TREE OF CONDITIONALS

The possible chains of conditionals form a tree, where each level is a generation step and each branch selects the next position. Training follows one path per sample, and sampling follows one path per image. We argue that paths that visit high-confidence positions first are preferable, for two reasons. *(i) Compounding errors.* Each conditional depends on all earlier choices. A low-likelihood choice early in the sequence moves the model into a poorly supported region of the tree, and subsequent predictions degrade. *(ii) No recovery mechanism.* Committed tokens are never revised. If an early token is implausible under the data distribution, the model conditions on out-of-distribution context for the remainder of generation, and errors cascade. Orders that begin with high-confidence tokens keep generation within high-density regions and thus reduce drift.

4 ORDERED AUTOREGRESSIVE SELF-DISTILLATION

Order specialization as capacity reallocation. An any-order model must fit conditionals for up to $N!$ permutations, which spreads its finite capacity over the entire permutation space. This generality is what enables zero-shot inpainting and editing. For unconstrained generation, however, only one good trajectory per image is needed. OAR exploits this asymmetry. It extracts a specialized order for each training image from the any-order model itself and fine-tunes the model on these orders. Capacity is thereby redirected toward the trajectories used at generation time, while the any-order conditionals are largely retained. The procedure has three stages.

4.1 STAGE 1: ANY-ORDER TRAINING

As in standard pipelines (Esser et al., 2021; Sun et al., 2024), images and their conditions (a class label or a caption) are mapped to discrete tokens by a VQ tokenizer. OAR is agnostic to the tokenizer. For any-order generation, the model must know *which* position it predicts next. We support two established mechanisms: *(a) RandAR-style position tokens* (Pang et al., 2025), where each content token is interleaved with a token that encodes the position of the next prediction; and *(b) σ-GPT-style double positional encoding* (Pannatier et al., 2024), where each token carries the encoding of its own position and that of the next position. In both cases, the model is trained with the standard next-token cross-entropy loss on random permutations.

4.2 Stage 2: Specialized order extraction

The trained any-order model labels every training image with a specialized order. Beam search over orders (Pannatier et al., 2024) is prohibitively expensive for $N{=}256$ tokens, so we use greedy selection based on the model's confidence. Under teacher forcing, at step i the model receives the ground-truth prefix $(\mathbf{x}_{<i}, \mathbf{l}_{<i})$ and predicts a distribution over the vocabulary V at every free location $l \in L_i$. The next position is the one that the model predicts most confidently. Scoring each location by its most likely token directly targets the failure mode of Section 3.2, namely committing to low-likelihood tokens early:

$$l_i^* = \arg\max_{l \in L_i} \max_{v \in V} p_\theta\big(v \mid l, \mathbf{x}_{<i}, \mathbf{l}_{<i}\big). \tag{3}$$

Repeating this for N steps yields a content-dependent order $\mathbf{l}^*$. The resulting pairs $(\mathbf{x}, \mathbf{l}^*)$ form a self-generated curriculum for the final stage. Extraction runs once, is embarrassingly parallel across images, and its output is cached and reused across fine-tuning runs.

4.3 Stage 3: Self-distillation

We fine-tune the any-order model on the pairs $(\mathbf{x}, \mathbf{l}^*)$. To avoid catastrophic forgetting of the any-order conditionals, we use a reduced learning rate and an order-annealing schedule in the spirit of RAR (Yu et al., 2024b). Training begins with random orders and gradually shifts toward the specialized orders. RAR anneals toward a single fixed raster order, whereas OAR anneals toward per-image orders that the model itself identified as easy to follow. Section 5.1 shows that this difference in the target order, rather than the annealing schedule, accounts for the improvement.

4.4 Inference

At test time we apply Eq. (3) with generated tokens in place of ground-truth tokens, so the model generates from its most to its least confident positions. A naive implementation requires $N(N{+}1)/2$ forward evaluations. Two optimizations make this practical (Appendices B.1 and B.2); for our text-to-image model, they bring the per-image latency close to that of a raster AR model (Table 9). *Parallel querying* evaluates all free positions in a single forward pass. The attention mask lets each candidate query attend to the committed history but not to the other candidates. *KV-cache overwriting* stores only committed position–content pairs, so the context matches training conditions and memory grows linearly in N. Replacing the $\arg\max$ in Eq. (3) with a top-k selection commits k tokens per step and thus enables parallel decoding (Section 5.4). OAR is also compatible with standard LLM inference optimizations such as batching, mixed precision, and FlashAttention.

Relation to MaskGIT. Confidence-ordered decoding resembles the iterative decoding of MaskGIT, but differs in two respects. First, OAR uses confidence to construct *training targets*: the extracted orders are distilled back into the model, whereas MaskGIT uses confidence only at test time. Second, OAR is a causal decoder, so committed tokens are cached and each step processes only the new queries, whereas MaskGIT's bidirectional encoder re-processes the full sequence at every step.

5 Experiments on Class-Conditional ImageNet

Setup. We evaluate on class-conditional ImageNet 256×256 (Deng et al., 2009). Our base model is the publicly released RandAR-XL checkpoint (Pang et al., 2025) (775M parameters, 300 epochs of any-order pretraining, $f{=}16$ VQ tokenizer, $N{=}256$ tokens). Self-distillation fine-tunes this checkpoint with AdamW at a learning rate of 10^{-5} with cosine decay, for 15k iterations at batch size 768 when order annealing is used. We report FID, Inception Score (IS), precision, and recall over 50k samples. All ImageNet samples, from OAR and from the RandAR checkpoints alike, use a classifier-free guidance scale of 3.4 and temperature 1.0. Unless stated otherwise, all models use 256 decoding steps. Full details are given in Appendix B.3.

Computational cost. OAR adds a small overhead to pretraining. Order extraction (Stage 2) is a one-time, embarrassingly parallel pass. Sharded across 4 nodes with 4 H100 GPUs each, and with no cross-node communication, it labels ImageNet in $\sim$1.5 hours. Self-distillation (Stage 3) takes $\sim$7

Table 1: **Effect of fine-tuning order, annealing, and sampling order** on ImageNet 256×256. All rows use RandAR-XL and 256 decoding steps. Annealing gradually shifts training from random orders to the fine-tuning order. The row marked † (random→raster) reproduces RAR's annealing (Yu et al., 2024b) on the same pretrained backbone and isolates the effect of the target order.

Fine-tuning	Annealing	Sampling	FID↓	IS↑	Prec.↑	Rec.↑
None (RandAR-XL)	–	Raster	12.31	127.55	0.59	0.70
	–	Greedy	3.64	220.48	0.61	0.74
	–	Random	2.39	253.91	0.80	0.60
Raster	✗	Random	6.57	160.18	0.70	0.65
	✗	Raster	2.56	236.15	0.79	0.60
	✓†	Raster	2.34	242.54	0.79	0.61
Greedy (OAR)	✗	Random	2.44	281.92	0.80	0.59
	✗	Greedy	2.22	256.77	0.80	0.60
	✓	Greedy	**2.17**	258.37	0.80	0.60

hours on a single 4-GPU node, which amounts to at most $\sim 5\%$ of the 300 pretraining epochs. For reference, pretraining corresponds to ~ 6 days of GPU running time on the same hardware.

5.1 Which order to fine-tune on and which to sample with

Table 1 compares three fine-tuning regimes (none, raster, and greedy), each evaluated under raster, greedy, and random sampling. Without fine-tuning, random sampling performs best (FID 2.39), whereas greedy (3.64) and especially raster (12.31) sampling perform worse. Uniform permutation training covers raster and greedy orders only with vanishing probability, and the statistics of these structured orders differ markedly from those of random ones. The any-order model therefore has no specific experience with them.

The target order, not the schedule, drives the gain. Fine-tuning on raster order reduces the number of orders from $N!$ to one, but it only matches the any-order baseline (2.56 without annealing, 2.34 with annealing). Reducing the number of orders is therefore not sufficient on its own; *which* order the model specializes to matters. The annealed raster row reproduces the component of RAR (Yu et al., 2024b) that is most closely related to our approach, namely random-to-raster annealing, on the same pretrained backbone and with the same training budget. Our random-to-greedy schedule improves on this control from 2.34 to 2.17 FID. The comparison holds the architecture, the annealing schedule, and the budget fixed. It also excludes RAR's orthogonal changes (scheduled guidance, AdaLN-Zero), so the gain is attributable to the specialized order itself. Finally, sampling the OAR model in *random* order still yields 2.44 FID, close to the 2.39 of the original model. Fine-tuning therefore leaves the any-order conditionals largely intact, which is the property that enables the zero-shot tasks in Section 5.3.

5.2 Comparison with the state of the art

Table 2 places OAR among models with fewer than 1B parameters. Self-distillation lowers the FID of RandAR-XL from 2.39 to 2.22, and to 2.17 with annealing. With the same parameter budget and backbone, OAR also outperforms the raster-order LlamaGen-XL (2.62). It improves on the causal any-order baseline and is competitive with fixed-order causal AR models of similar size. RAR-XL attains a lower FID, but it converges to a single raster order. It also combines order annealing with the orthogonal architectural and guidance changes discussed above, and it forfeits the any-order conditionals that inpainting and outpainting require. Among the any-order models in Table 2, OAR improves on its RandAR baseline and on MaskGIT (4.02). Bidirectional MAR models achieve lower generation FID, but they are incompatible with standard KV caching and, as shown next, perform worse on zero-shot inpainting and outpainting.

Table 2: **Class-conditional ImageNet** 256×256 for models with fewer than 1B parameters. RandAR rows are our evaluation of the publicly released RandAR checkpoints under the same protocol as OAR; all other baselines are as reported by Pang et al. (2025). Under the original authors' evaluation setup, the published RandAR-XL result is 2.22 FID at 256 steps. *Trained at 384×384 and resized to 256×256. "-re": rejection sampling. *Any order*: the same model supports arbitrary-order generation, and hence zero-shot inpainting and outpainting.

Type	Model	#Params	FID↓	IS↑	Prec.↑	Rec.↑	Steps	Any order
GAN	BigGAN (Brock et al., 2019)	112M	6.95	224.5	0.89	0.38	1	✗
	GigaGAN (Kang et al., 2023)	569M	3.45	225.5	0.84	0.61	1	✗
	StyleGAN-XL (Sauer et al., 2022)	166M	2.30	265.1	0.78	0.53	1	✗
Diffusion	ADM (Dhariwal & Nichol, 2021)	554M	4.59	186.7	0.82	0.52	250	✗
	LDM-4 (Rombach et al., 2022)	400M	3.60	247.7	–	–	250	✗
	DiT-XL (Peebles & Xie, 2023)	675M	2.27	278.2	0.83	0.57	250	✗
	SiT-XL (Ma et al., 2024)	675M	2.06	270.3	0.82	0.59	250	✗
Bidirectional (masked)	MaskGIT-re (Chang et al., 2022)	227M	4.02	355.6	–	–	8	✓
	TiTok-S-128 (Yu et al., 2024c)	287M	1.97	281.8	–	–	64	✓
	MAR-L (Li et al., 2024)	479M	1.98	290.3	–	–	64	✓
	MAR-H (Li et al., 2024)	943M	1.55	303.7	0.81	0.62	256	✓
Causal AR	VAR (Tian et al., 2024)	600M	2.57	302.6	0.83	0.56	10	✗
	SAR-XL (Liu et al., 2024)	893M	2.76	273.8	0.84	0.55	256	✗
	RAR-B (Yu et al., 2024b)	261M	1.95	290.5	0.82	0.58	256	✗
	RAR-L (Yu et al., 2024b)	461M	1.70	299.5	0.81	0.60	256	✗
	RAR-XL (Yu et al., 2024b)	955M	1.50	306.9	0.80	0.62	256	✗
	LlamaGen-L (Sun et al., 2024)	343M	3.07	256.1	0.83	0.52	256	✗
	LlamaGen-XL* (Sun et al., 2024)	775M	2.62	244.1	0.80	0.57	576	✗
	RandAR-L (Pang et al., 2025)	343M	2.65	249.0	0.82	0.56	256	✓
	RandAR-XL (Pang et al., 2025)	775M	2.39	253.9	0.80	0.60	256	✓
	OAR (ours, w/o annealing)	775M	2.22	256.8	0.80	0.60	256	✓
	OAR (ours, w/ annealing)	775M	**2.17**	258.4	0.80	0.60	256	✓

Table 3: **Zero-shot inpainting and outpainting** FID↓ on ImageNet 256×256 with random masks covering 25%–75% of the image. We evaluate all methods ourselves with identical masks and protocol, including MaskGIT (Chang et al., 2022) and MAR (Li et al., 2024). *Causal*: decoder-only with standard KV caching.

Method	Causal	Inpaint.	Outpaint.
MaskGIT	✗	8.53	10.33
MAR	✗	5.63	3.62
RandAR-XL	✓	2.14	3.21
OAR (ours)	✓	**2.10**	**3.17**

Table 4: **Parallel decoding.** FID of OAR on ImageNet 256×256 versus the total number of decoding steps, obtained by committing the top-k most confident positions per step.

Steps	FID↓
256 (k=1)	2.17
128	2.25
88	2.51
64	3.04

5.3 Zero-shot inpainting and outpainting

A key motivation for OAR is to improve quality *without* sacrificing the flexibility of any-order generation. For inpainting, we initialize the context with the visible tokens and let the model generate the masked region in its own greedy order. Table 3 shows that OAR slightly improves over its any-order baseline in both inpainting (2.10 vs. 2.14 FID) and outpainting (3.17 vs. 3.21). Its inpainting precision (0.85 vs. 0.79) and recall (0.63 vs. 0.60) are also higher. Self-distillation therefore does not collapse the model onto a single rigid trajectory. OAR also outperforms the bidirectional MaskGIT (8.53 / 10.33) and MAR (5.63 / 3.62) on both tasks, while remaining a causal decoder with standard KV caching. Figure 1 shows qualitative inpainting and class-conditional editing results.

5.4 Parallel decoding

Because OAR inherits RandAR's any-order conditionals, it can commit several tokens per step by replacing the $\arg\max$ in Eq. (3) with a top-k selection. Table 4 shows that halving the number of steps to 128 costs only 0.08 FID, whereas more aggressive schedules incur larger losses (2.51 at 88

Table 5: **Fashion Products.** Training and generation orders. d: mean Euclidean distance between consecutively generated patches. IS is close to 1 because ImageNet classes carry little information for single-product images; we report it for completeness.

Train	Gen.	FID↓	IS↑	d
Raster	Raster	4.58	1.106	1.83
Random	Raster	4.38	1.102	1.83
Random	Random	4.07	1.103	8.19
OAR (ours)	Greedy	**2.56**	**1.111**	3.99

Table 6: **Multimodal CelebA-HQ.** Text-to-image FID, all evaluated by us on the same test split and protocol, compared with TediGAN (Xia et al., 2021) and Stable Diffusion v2 (Rombach et al., 2022) fine-tuned with LoRA (Hu et al., 2022).

Model	FID↓
TediGAN	8.27
SD v2 + LoRA	2.47
Raster AR	1.94
Any-order + greedy	1.52
OAR (ours)	**1.41**

steps and 3.04 at 64 steps). Parallel decoding is not the focus of this work. OAR retains the capability without any architectural change and is complementary to methods designed specifically for few-step generation.

5.5 Human evaluation

FID measures distribution-level statistics and can miss localized artifacts. We therefore ran a two-alternative forced-choice study between OAR and RandAR-XL. Participants first gave informed consent. They then compared 21 image pairs, one for each of 21 randomly sampled ImageNet classes. The pairs were generated from random seeds without curation, with matched seeds, classifier-free guidance, and temperature, and with randomized left/right placement. In total, 51 raters provided 1,071 judgments. OAR was preferred in **64.3%** of judgments. It won the majority vote in 16 of the 21 pairs (Figure 3) (two-sided sign test over pairs, $p = 0.027$) and received more than 65% of the votes in 11 of them. The improvement in FID thus corresponds to a perceptible improvement in visual quality. The full protocol and per-class results appear in Appendix B.5.

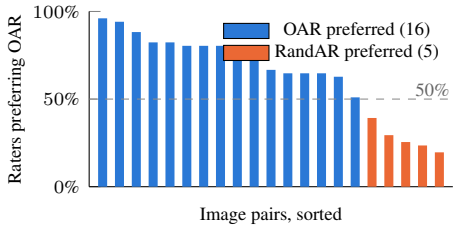

Figure 3: **Per-pair human preference** for OAR over RandAR-XL (51 raters per pair).

6 Transfer to Text Conditioning and a Second Architecture

Setup. To test whether OAR transfers beyond class conditioning and beyond RandAR, we train a *different*, scratch-trained any-order decoder on text-to-image generation. This decoder uses σ-GPT-style double positional encoding in which the next position is encoded *relative* to the current one (Appendix C.4). The model is a 6-layer transformer (width 768) over a frozen VQGAN (Esser et al., 2021) with a 1,024-entry codebook and 16×16 patches. It is trained for 300 epochs of any-order training and then self-distilled for 150 epochs; architecture and training details are given in Appendix C.1. At inference, this model selects locations with a distance-regularized variant of Eq. (3) that penalizes large jumps from the previous location, with weight $\lambda = 0.5$ (Appendix C.7). Unless stated otherwise, all ordered text-to-image results use this rule. Its effect is small (3.15 vs. 3.02 FID on Fashion Products; Table 10). We use Fashion Products (Aggarwal, 2021) (44.4k product images with captions) and Multimodal CelebA-HQ (Xia et al., 2021) (30k faces, each with several attribute captions), both split 90/10 into training and test sets. We report FID and IS, together with the mean Euclidean distance d (in patches) between consecutively generated patches. Compared with ImageNet, these datasets have more controlled distributions, which isolates the effect of specialized orders from real-world variability.

Fashion Products. Table 5 reproduces the known advantage of random-order training over raster order (Li et al., 2024; Fan et al., 2024; Yu et al., 2024b) (4.07 vs. 4.58 FID). OAR, which generates in a self-distilled greedy order, reduces the FID to 2.56 and also improves IS. Removing the self-distillation stage while keeping the same total number of epochs raises the FID to 2.98 (Appendix C.9),

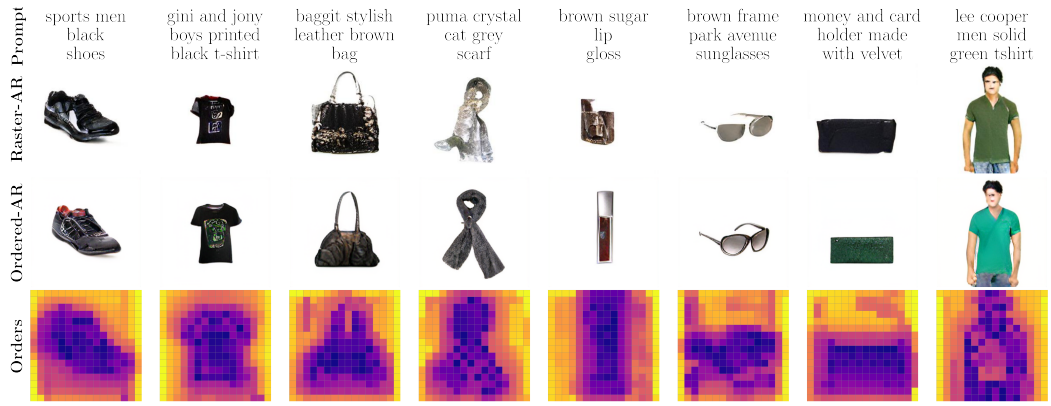

Figure 4: **Fashion Products samples. (Top)** Raster AR. **(Middle)** OAR. **(Bottom)** OAR generation order, from yellow (early) to violet (late). The specialized orders follow the image content and often yield cleaner samples.

which confirms that the gain comes from distillation rather than from longer training. Figure 4 shows that the extracted orders are closely correlated with image content.

Multimodal CelebA-HQ. Table 6 shows the same trend: greedy ordering improves over raster order (1.94 $\rightarrow$ 1.52), and self-distillation improves the FID further to 1.41. OAR also outperforms TediGAN (8.27) and Stable Diffusion v2 fine-tuned with LoRA (2.47). The specialized orders have a semantically structured progression. The model first generates high-confidence facial regions such as the cheeks and chin, proceeds to fine-grained features such as the eyes, lips, and hair, and finishes with the background. It tends to complete one semantic region before moving to the next, which decomposes generation into coherent sub-problems. This is consistent with the capacity-reallocation view, since each conditional is evaluated with semantically coherent context. Additional samples and an analysis of background effects appear in Appendices C.6 and C.10.

7 Conclusion

We presented Ordered Autoregressive (OAR) generation, a self-distillation procedure that extracts specialized, content-dependent generation orders from an any-order AR model's own confidence scores and fine-tunes the model on them. Order specialization reallocates capacity from the full permutation space to the trajectories that are actually used at generation time. It improves sample quality on ImageNet, Fashion Products, and Multimodal CelebA-HQ, and human raters confirm the improvement. At the same time, OAR preserves the zero-shot inpainting, outpainting, editing, and parallel decoding abilities of the any-order model. A controlled comparison with random-to-raster annealing on the same backbone shows that the gain stems from the specialized order itself rather than from the training schedule. On ImageNet, OAR requires only a short fine-tuning stage (at most $\sim$5% of the pretraining epochs), no architectural change, and no additional annotation, and it remains a causal decoder that is compatible with the LLM inference stack.

Limitations and future work. Our experiments are limited to models below 1B parameters. Competitive open-domain text-to-image systems typically have 3B or more parameters, and training them requires multi-node sharded infrastructure beyond our compute budget. The procedure itself is architecture-agnostic, and we view scaling OAR as a natural next step. Similarly, our text-to-image evaluation uses curated datasets rather than open-domain prompts. On ImageNet, we apply OAR to a single any-order backbone (RandAR). Our text-to-image experiments provide evidence of transfer to a second, independently trained architecture, but applying OAR to other any-order decoders remains to be tested. Greedy order selection adds inference overhead, which parallel querying and KV-cache overwriting largely offset. Learning an explicit order predictor could remove this overhead altogether. Finally, alternating order extraction and fine-tuning over several rounds may further refine the orders.

AI USE STATEMENT

In this work, we used generative AI tools to edit the manuscript for readability, to reformat it to the ICLR 2027 template, and to format references. We did not use generative AI tools to generate synthetic data, develop theoretical models or frameworks, formulate mathematical claims or proofs, propose or refine hypotheses, design experiments, implement methods, clean or reformat datasets, analyze qualitative data, or interpret results. All experimental results were produced by the authors' own code and experiments. We reviewed all AI-assisted text against our experimental records, and we take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work includes a human preference study (Section 5.5 and Appendix B.5), conducted with institutional review board (IRB) clearance. Participation was voluntary and preceded by an informed-consent statement. Participants could withdraw at any time without penalty and received no compensation, and no personal information was collected from participants who declined. All datasets we use (ImageNet, Fashion Products, and Multimodal CelebA-HQ) are publicly available research datasets, and we use them under their respective terms. Multimodal CelebA-HQ contains images of real people. As with any image generator, and face generators in particular, models of this kind could be misused to produce misleading content, and they may reproduce biases present in their training data. Our models are small research-scale models trained on public benchmarks, and we do not release any system intended for deployment.

REPRODUCIBILITY STATEMENT

Our ImageNet experiments start from the publicly released RandAR-XL checkpoint (Pang et al., 2025). The order extraction rule is given in Eq. (3), and the fine-tuning hyperparameters, tokenizer, evaluation protocol, and hardware are listed in Appendix B.3. The parallel-query attention mask and the KV-cache overwriting scheme used for efficient inference are specified in Appendices B.1 and B.2. The text-to-image architecture, positional encodings, training schedule, and generation procedure are described in Appendices C.1, C.2 and C.4, including pseudo-code in Algorithms 1 and 2. The human-study protocol and per-class results are reported in Appendix B.5. We will release our code upon publication.

REFERENCES

Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant Nair, Ilya Soloveychik, and Purushotham Kamath. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. In *MLSys*, volume 7, 2024.

Param Aggarwal. Fashion product images dataset, 2021. URL https://www.kaggle.com/datasets/paramaggarwal/fashion-product-images-dataset.

Anthropic. The claude 3 model family: Opus, sonnet, haiku, 2024. URL https://api.semanticscholar.org/CorpusID:268232499.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *CVPR*, pp. 11315–11325, 2022.

Mark Chen, Alec Radford, Rewon Child, Jeff Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, 2020.

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

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *ICLR*, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pp. 4171–4186, 2019.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Philipp Dufter, Martin Schmitt, and Hinrich Schütze. Position information in transformers: An overview. *Computational Linguistics*, 48(3):733–763, 09 2022. ISSN 0891-2017. doi: 10.1162/coli_a_00445. URL https://doi.org/10.1162/coli_a_00445.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, pp. 12873–12883, 2021.

Lijie Fan, Tianhong Li, Siyang Qin, Yuanzhen Li, Chen Sun, Michael Rubinstein, Deqing Sun, Kaiming He, and Yonglong Tian. Fluid: Scaling autoregressive text-to-image generative models with continuous tokens. *arXiv preprint arXiv:2410.13863*, 2024.

Nicolas Ford, Daniel Duckworth, Mohammad Norouzi, and George E Dahl. The importance of generation order in language modeling. *arXiv preprint arXiv:1808.07910*, 2018.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive KV cache compression for LLMs. In *ICLR*, 2024.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *ICML*, pp. 881–889, 2015.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics*, 7:661–676, 2019.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, pp. 544–560, 2020.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers, 2020.

Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. In *ICLR*, 2022.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Iris AM Huijben, Wouter Kool, Max B Paulus, and Ruud JG Van Sloun. A review of the gumbel-max trick and its extensions for discrete stochasticity in machine learning. *IEEE transactions on pattern analysis and machine intelligence*, 45(2):1353–1371, 2022.

Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10124–10134, 2023.

Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *ICML*, pp. 3499–3508, 2019.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *ACM SIGOPS*, 2023.

Tianhong Li, Huiwen Chang, Shlok Mishra, Han Zhang, Dina Katabi, and Dilip Krishnan. Mage: Masked generative encoder to unify representation learning and image synthesis. In *CVPR*, pp. 2142–2152, 2023.

Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. In *NeurIPS*, 2024.

Xuanlin Li, Brandon Trabucco, Dong Huk Park, Michael Luo, Sheng Shen, Trevor Darrell, and Yang Gao. Discovering non-monotonic autoregressive orderings with variational inference. In *ICLR*, 2021.

Wenze Liu, Le Zhuo, Yi Xin, Sheng Xia, Peng Gao, and Xiangyu Yue. Customize your visual autoregressive recipe with set autoregressive modeling. *arXiv preprint arXiv:2410.10511*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pp. 23–40. Springer, 2024.

Shikib Mehri and Leonid Sigal. Middle-out decoding. In *NIPS*, volume 31, 2018.

OpenAI. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

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 *CVPR*, 2025.

Arnaud Pannatier, Evann Courdier, and François Fleuret. σ-gpts: A new approach to autoregressive models. In *ECML PKDD*, pp. 143–159, 2024.

Vassilis Papadopoulos, Jérémie Wenger, and Clément Hongler. Arrows of time for large language models. *arXiv preprint arXiv:2401.17505*, 2024.

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

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In *MLSys*, volume 5, pp. 606–624, 2023.

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

Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *NIPS*, volume 32, 2019.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *ACM SIGGRAPH 2022 conference proceedings*, pp. 1–10, 2022.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *NAACL*, 2018.

Andy Shih, Dorsa Sadigh, and Stefano Ermon. Training and inference on any-order autoregressive models the right way. In *NIPS*, volume 35, pp. 2762–2775, 2022.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation, 2024. URL https://arxiv.org/abs/2406.06525.

Qing Sun, Stefan Lee, and Dhruv Batra. Bidirectional beam search: Forward-backward inference in neural sequence models for fill-in-the-blank image captioning. In *CVPR*, pp. 6961–6969, 2017.

Gemini Team and Google. Gemini: A family of highly capable multimodal models, 2024. URL https://arxiv.org/abs/2312.11805.

Qwen Team and Alibaba Group. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671.

Keyu Tian, Yi Jiang, Zehuan Yuan, BINGYUE PENG, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. In *International Conference on Machine Learning*, pp. 467–475. PMLR, 2014.

Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *JMLR*, 17(205):1–37, 2016.

Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In *NIPS*, volume 29, 2016.

Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, pp. 1747–1756, 2016.

Aaron Van Den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NIPS*, volume 30, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, volume 30, 2017.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. In *ICLR*, 2016.

Xinyi Wang, Hieu Pham, Pengcheng Yin, and Graham Neubig. A tree-based decoder for neural machine translation. In *EMNLP*, 2018.

Zhe Wang, Jiaxin Shi, Nicolas Heess, Arthur Gretton, and Michalis K. Titsias. Learning-order autoregressive models with application to molecular graph generation, 2025. URL https://arxiv.org/abs/2503.05979.

Sean Welleck, Kianté Brantley, Hal Daumé Iii, and Kyunghyun Cho. Non-monotonic sequential text generation. In *ICML*, pp. 6716–6726, 2019.

Weihao Xia, Yujiu Yang, Jing-Hao Xue, and Baoyuan Wu. Tedigan: Text-guided diverse face image generation and manipulation. In *CVPR*, 2021.

Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A theory of usable information under computational constraints. In *ICLR*, 2020.

Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *ACL*, pp. 523–530, 2001.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NIPS*, volume 32, 2019.

Lijun Yu, Jose Lezama, Nitesh Bharadwaj Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David A Ross, and Lu Jiang. Language model beats diffusion - tokenizer is key to visual generation. In *ICLR*, 2024a.

Qihang Yu, Ju He, Xueqing Deng, Xiaohui Shen, and Liang-Chieh Chen. Randomized autoregressive visual generation. *arXiv preprint arXiv:2411.00776*, 2024b.

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

Distilling Specialized Orders for Visual Generation

Supplementary Material

Table of Contents

- A Additional Related Work
  - A.1 Orders in Autoregressive Models
  - A.2 Positional Encodings and Orders
- B Class-to-Image
  - B.1 Parallel Query Evaluations via Masking
  - B.2 Optimized KV Caching via Overwriting
  - B.3 ImageNet Experimental Setup
  - B.4 Qualitative Samples
  - B.5 Human Evaluation Protocol, Study Design, and Results
- C Text-to-Image
  - C.1 Implementation Details
  - C.2 Generation Process
  - C.3 V-information
  - C.4 Different Generation Approaches
  - C.5 KV Caching
  - C.6 Effect of Background
  - C.7 Distance Regularization
  - C.8 Decoding Strategies
  - C.9 Impact of Self-Distillation
  - C.10 Qualitative Samples on Multimodal CelebA-HQ

A Additional related work

A.1 Orders in Autoregressive Models

Research has shown that the order in which we organize input and/or output data in the sequence-to-sequence framework matters significantly when learning an underlying model (Vinyals et al., 2016). Such insights have led to the development of Any-Order Autoregressive (AO-AR) Models, in which the autoregressive model does not impose a fixed order of generation. Some methods (Germain et al., 2015; Uria et al., 2016) train a single deep feed-forward neural network that can assign a conditional distribution to any variable given any subset of the others. During inference, they create an ensemble of models by sampling a set of orderings, computing the likelihood under each order, and averaging, thus baking an "orderless" inductive bias into the model. Further work (Shih et al., 2022) refines the space of orderings used during training to avoid the redundancy present in previous probabilistic models by training on a smaller set of univariate conditionals that still maintain support for efficient arbitrary conditional inference. In the continuation of AO-ARM approaches, more research has been conducted.

In Natural Language Processing (NLP) specifically, motivated by enabling AR language models to learn bidirectional contexts, XLNet (Yang et al., 2019) maximizes the expected log-likelihood over all permutations of the factorization order by relying on a proper attention mechanism in Transformers. σ-GPT (Pannatier et al., 2024) learns a model that can generate sequences of text tokens in any order and modulates the generation order on the fly per-sample during inference in a fashion already explored for Neural Architecture Search (Guo et al., 2020).

A.2 Positional Encodings and Orders

Since the self-attention mechanism in transformers is inherently permutation-invariant, it does not account for token order. It requires additional overhead to encode positional information (Vaswani et al., 2017; Dufter et al., 2022). Early methods introduce absolute positional embeddings, encoded through sinusoidal functions or learnable embeddings, to differentiate tokens based on their positions rather than just their content (Dosovitskiy, 2020; Vaswani et al., 2017). However, as the number of tokens grows, absolute positional embeddings become less efficient. To address this limitation, Shaw et al. (2018) introduce relative position biases directly into the attention matrix within the layers of self-attention, enhancing spatial awareness and scalability. Although both absolute- and relative position embeddings are effective in fixed-token-size scenarios, they struggle with adapting to variable token sizes and require flexibility and extrapolation capabilities. To overcome these challenges, Su et al. (2024) propose an approach that encodes absolute positions using a rotation matrix and integrates explicit relative positional dependencies into the self-attention formulation. This combined method enhances the adaptability to different resolutions, enabling the model to generalize better across varying spatial contexts.

In our work, following the same motivation as (Gu et al., 2019), we leverage relative positions for our text-to-image setting to alleviate the challenges of predicting absolute positions. Our method introduces two key novelties: First, we extend the relative positional encoding to handle 2D inputs effectively. Second, we employ a dual positional encoding scheme—preserving the positional encoding of the current patch as absolute, while encoding the position of the next patch relative to the current one. Notably, our proposed relative positional embedding differs from prior methods, where relative positions are directly embedded within the transformer architecture itself (Yang et al., 2019). This positional encoding strategy is used throughout OAR's any-order AR backbone for the text-to-image setting.

B Class-to-Image

For class conditional training on ImageNet (Deng et al., 2009), we use the architecture from RandAR (Pang et al., 2025) which is optimized for large-scale training. Specifically, we use the publicly released RandAR-XL checkpoint.

B.1 Parallel Query Evaluations via Masking

To select the next token location, we evaluate K candidate positions in parallel. Let t be the number of previously generated (committed) tokens. In a standard autoregressive setup, appending K queries would cause the i-th query to attend to the j-th query (where $j < i$), creating a dependency between the candidates that should be evaluated independently.

To batch these evaluations into a single forward pass, we modify the causal attention mask $\mathbf{A}$. For a sequence containing t history tokens and K query tokens, the mask $\mathbf{A} \in \{0,1\}^{(t+K)\times(t+K)}$ is constructed as follows:

$$\mathbf{A}_{i,j} = \begin{cases} 1 & \text{if } i \le t \text{ and } j \le i \quad \text{(Causal History)} \\ 1 & \text{if } i > t \text{ and } j \le t \quad \text{(Queries see History)} \\ 1 & \text{if } i = j \qquad\qquad\quad \text{(Self-Attention)} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

This structure ensures that while all queries can attend to the full history $(1 \dots t)$, the block representing query to query interactions $(i, j > t)$ is strictly diagonal. This makes the distribution for each query independent of previous queries. Figure 5 illustrates this mask for a scenario with 7 tokens ($t = 4$ history tokens and $K = 3$ parallel queries).

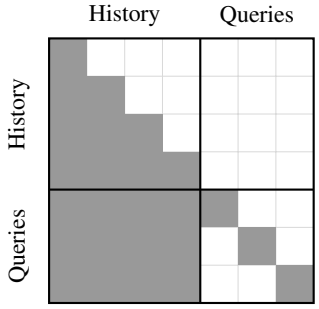

Figure 5: **Independent Query Attention Mask.** An example for a 7 token image ($t = 4$ committed tokens and $K = 3$ candidate queries). Gray cells ($\mathbf{A}_{i,j} = 1$) indicate allowed attention; white cells ($\mathbf{A}_{i,j} = 0$) are masked. The bottom-right block is strictly diagonal, preventing queries from attending to each other.

B.2 Optimized KV Caching via Overwriting

Standard KV caching appends new tokens to the end of the cache at every step. In our framework, we evaluate K temporary queries but only commit a subset (e.g., 1 position-content pair of tokens) to the sequence. We employ a cache overwriting strategy that reuses memory indices.

Context Consistency. The primary goal of this optimization is to ensure the model attends to a clean context. During training, the model sees a sequence of image tokens interleaved with their corresponding position token. If we simply appended all K evaluated queries to the cache, the context for the next step would be polluted with $K - 1$ rejected positions. This would shift the distribution away from training conditions. By overwriting the cache indices of the queries with the token(s) for the selected position(s), we ensure the next generation step attends only to the committed path, preserving the structure of the context.

Memory Complexity. Without overwriting, the cache would accumulate every candidate ever evaluated. In a generation process of length N where the number of candidates K_i decreases as the image fills up (i.e., $K_i \approx N - i$), the total cache size would grow to $\sum_{i=0}^{N}(N - i) \approx O(N^2)$. By overwriting the temporary queries with the committed tokens, the cache grows linearly with the sequence length $2N$ (considering interleaved position-content pairs), maintaining a memory complexity of $O(N)$.

B.3 IMAGENET EXPERIMENTAL SETUP

Base Model. All experiments use a RandAR-XL (Pang et al., 2025) model. We use the publicly available pretrained checkpoint as our any-order model.

Tokenizer. We use a pre-trained VQGAN tokenizer with a codebook size of $16,384$, an embedding dimension of 8, and a downsampling factor of $f = 16$. For a 256×256 input image, this yields the 256 discrete tokens used by the autoregressive model.

Training. For the self-distillation phase, we fine-tune the pre-trained any-order model using the extracted specialized orders. We use the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a weight decay of 0.05. To ensure training stability, we apply gradient clipping with a maximum norm of 1.0. For the fine-tuning annealing experiments, we train on the ImageNet training set for a total of 15000 steps with a global batch size of 768 (≈ 9 epochs). For experiments involving order annealing, we set the annealing start and end to 1 and 3 epochs, respectively. For the experiments without annealing, we train for a total of 5000 steps with a global batch size of 768 (≈ 3 epochs). The learning rate is set to 1×10^{-5} with a cosine decay schedule without a warm-up phase.

Evaluation. We report Fréchet Inception Distance (FID), Inception Score (IS), Precision, and Recall on the ImageNet (Deng et al., 2009) validation set using 50,000 generated samples. All models, including the RandAR baselines, are sampled with a classifier-free guidance scale of 3.4 and temperature 1.0.

Hardware. For the ImageNet experiments, we use compute nodes with 4×NVIDIA H100 SXM5 GPUs (80 GB each), 32 AMD EPYC 9454 CPU cores, and 128 GB of RAM. Order extraction is sharded across 4 such nodes without cross-node communication and takes $\sim$1.5 hours for the full ImageNet training set; self-distillation takes $\sim$7 hours on a single node.

B.4 QUALITATIVE SAMPLES

Figure 6 shows additional ImageNet samples together with their specialized generation orders.

B.5 HUMAN EVALUATION PROTOCOL, STUDY DESIGN, AND RESULTS

We describe the steps we took to conduct the human evaluation study, the ethical considerations involved, and the detailed outcomes. All participants were first shown a consent statement explaining that their participation was voluntary, that they could stop at any time without penalty, and that their responses were provided freely without any external pressure or influence. The statement also clarified that the study did not involve any monetary or non-monetary compensation. Only participants who agreed to these terms were allowed to continue. Those who declined were shown a short thank-you message and exited the study immediately, and we did not collect any information from them.

Participants who consented proceeded to a series of image-pair comparison questions. In each pair, one image was generated by our OAR model and the other by the RandAR-XL baseline. To keep the evaluation fair, we randomized the left–right placement of the images for every participant. Both images in a pair are generated from the same class condition, random seed, sampling temperature, and classifier-free guidance scale, so that no model is favored by the sampling configuration.

Table 7 summarizes the per-question human preference rates across all 21 evaluation pairs sampled randomly. Each percentage represents the fraction of participants who selected the OAR-generated image as the better one for that specific pair. While preference levels vary across categories—reflecting the diversity and difficulty of the ImageNet classes—the overall trend is consistent: in most cases, participants preferred the outputs of our model over those of the baseline. Five questions (Q10, Q12, Q14, Q15, Q17) favor the baseline, which reflects the difficulty of certain fine-grained ImageNet classes; OAR is preferred in the remaining 16, and by more than 65% of raters in 11 of them.

Averaged across all 21 questions, our method achieved a preference rate of **64.33%**. This aligns with the aggregate human preference reported in the main paper and reinforces that the improvements captured by automated metrics correspond to perceptible gains in visual quality.

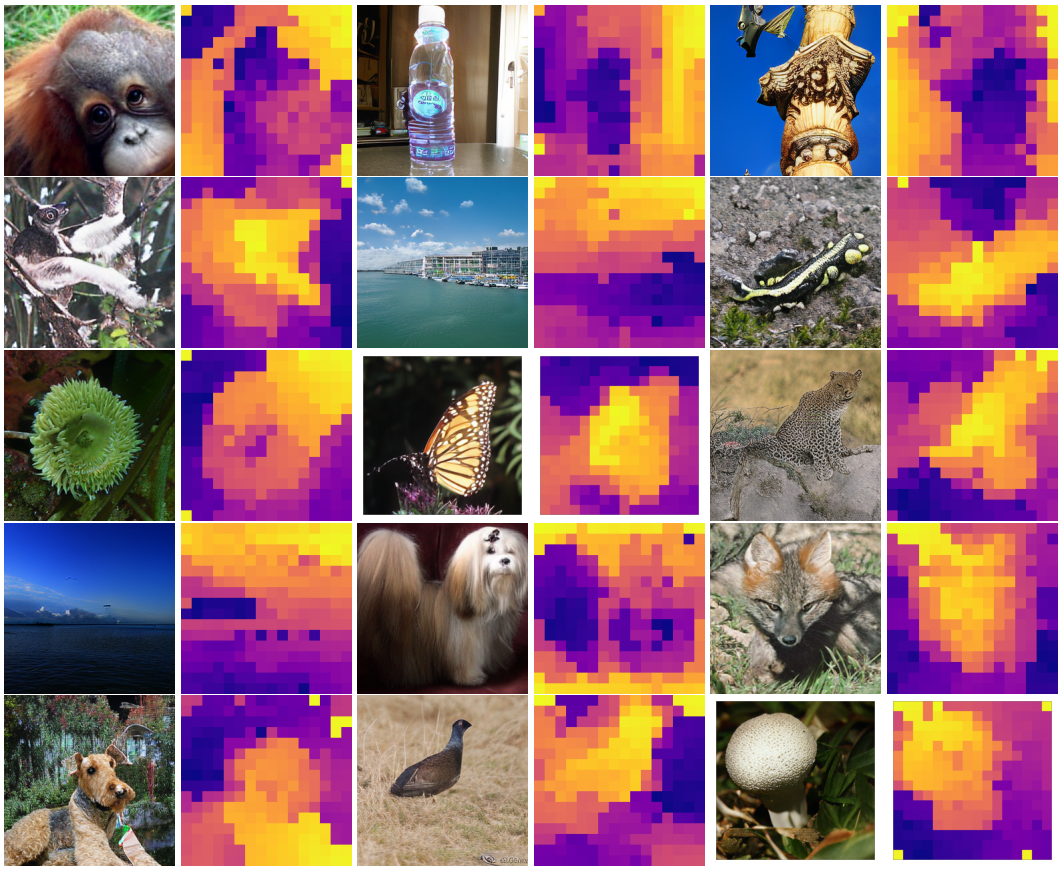

Figure 6: **Visualization of Images and Specialized Orders.** Examples of generated images with corresponding generation order heatmaps. Lighter areas represent patches generated earlier by the model. The visualization shows that the model does not follow a fixed heuristic (e.g., always background first). After self-distillation, the model prioritizes high-confidence regions—whether homogeneous backgrounds (e.g., clear skies) or salient features (e.g., the goldfish)—before tackling more ambiguous regions.



Figure 7: The **top row** shows samples generated by our OAR model, while the **bottom row** displays the corresponding outputs from the RandAR-XL (Pang et al., 2025) baseline.

Table 7: Human preference rates (%).

Q1	Q2	Q3	Q4	Q5	Q6	Q7
64.71	76.47	88.24	50.98	64.71	96.08	94.12
Q8	Q9	Q10	Q11	Q12	Q13	Q14
80.39	78.43	23.53	66.67	19.61	64.71	25.49
Q15	Q16	Q17	Q18	Q19	Q20	Q21
39.22	62.75	29.41	82.35	80.39	80.39	82.35
Average				**64.33**		

Example image pairs used in our human evaluation study are shown in Fig. 7. These pairs represent the comparison task that participants completed, where they selected whichever image they felt looked better overall, whether in terms of realism, coherence, or visual appeal.

C TEXT-TO-IMAGE

C.1 IMPLEMENTATION DETAILS

For the text-to-image experiments (Section 6), we train a single decoder-only transformer from scratch. Images are encoded with a pretrained VQGAN (Esser et al., 2021) encoder with a 16×16 patch size and a codebook of 1,024 entries, and the matching VQGAN decoder maps tokens back to pixels. All models share the same configuration. For each image, one caption is sampled at random from the captions provided by the dataset. The text sequence is truncated or padded to 256 tokens and concatenated with the image-token sequence. The transformer has an embedding dimension of 768, depth 6, and 14 attention heads of dimension 64. Positional embeddings are learnable and additively factorized for parameter efficiency (Ho et al., 2020). We use layer normalization (Ba et al., 2016) and apply dropout (Srivastava et al., 2014) with $p = 0.2$ to the linear projections of both the feed-forward and self-attention layers. Images are augmented with random resized crops. Following Ramesh et al. (2021), the loss on image tokens is weighted by a factor of 7 relative to the loss on text tokens. We use AdamW (Loshchilov & Hutter, 2019) with a learning rate of 3×10^{-4}, which is reduced by a factor of 0.8 on plateau. Models are trained for 300 epochs of any-order training with a batch size of 16 per machine, followed by self-distillation on the extracted orders (Appendix C.9).

C.2 GENERATION PROCESS

We provide additional details on how our approach (OAR) performs in text conditioned image generation through generation pipeline in Algorithm 1. During inference, the transformer receives text embeddings as conditioning inputs and autoregressively predicts a sequence of discrete VQ-GAN (Esser et al., 2021) token indices. So that every location can be queried, including the first one, we prepend a zero token to the image sequence (Pannatier et al., 2024).

To enable efficient sampling through parallelization, we use batched processing wherein each batch dimension computes one location. We sample codebook indices with the Gumbel-Top-k trick (Kool et al., 2019), using standard Gumbel noise ($\mu = 0$, $\beta = 1$) (Huijben et al., 2022).

Since we keep the pretrained VQGAN-VAE frozen, the spatial structure of the output must be restored after generation. We therefore record the model's chosen patch locations throughout sampling and subsequently reorder the predicted tokens before decoding them into the final image. Algorithm 1 summarizes the full inference procedure.

C.3 V-INFORMATION

In this section, we also examine the impact of order from an information-theoretic perspective (Shannon, 1948) concerning the text conditioned image generation. When ignoring computational considerations, different factorizations of conditional probability should theoretically yield equivalent results. However, traditional information theory often overlooks essential computational factors relevant to practical applications. To address these nuances, we apply $\mathcal{V}$-information (Xu et al., 2020), which intuitively quantifies the additional information about a random variable X that can be extracted from

Algorithm 1 Generation Process

Input: Condition parameters, the generative AR transformer engine t_Ω, decoder d_ψ
Output: A generated image based on the given condition parameters

```
Initialize a list of possible positions P(x, y) of total length n
Compute reference absolute (emb^A) and relative (emb^R) embeddings
Predict the first token using condition parameters with t_Ω
for i = (1, 2...n) AR steps do
    Extract the relative positions from (P) for all locations
    Get emb^R for the relative positions
    Replicate the absolute positions for n − i + 1 times
    Retrieve emb^A for the absolute positions
    Concatenate emb^A and emb^R
    Compute in parallel for n − i + 1 positions
    Select the most favorable patch location (Eq. (8)) l* and record it
    Use Gumbel-Top-k sampling trick for the best codebook index
    Remove l* from P
end for
Discard the first (zero) token
Reorder the patches using recorded locations
Decode the patches tokens using d_ψ
```

another variable Y using any predictor in $\mathcal{V}$. We define $\mathcal{V}$-entropy as the uncertainty we have in Y after observing X as follows:

$$H_{\mathcal{V}}(Y|X) = \inf_{f \in \mathcal{V}} E[-\log f[x](y)], \tag{5}$$

where $f[x](y)$ produces a probability distribution over the tokens. Information terms like $I_{\mathcal{V}}(X \to Y)$ are defined analogous to Shannon information, that is, $I_{\mathcal{V}}(X \to Y) = H_{\mathcal{V}}(Y) - H_{\mathcal{V}}(Y|X)$. In our setup, we define $\mathcal{V}$ as the Any-Order Autoregressive Model family of functions, described in Section 3.1, where Y denotes the next token and X represents the context tokens. These context tokens are arranged based on two distinct ordering schemes: $\mathbf{l^0}$ for the raster-scan order and $\mathbf{l}^\tau$ for the learned order inferred by our model. We define cross-entropy as $H_{\mathcal{V}}(\cdot)$, which we use to measure $\mathcal{V}$-information. Since the next tokens differ between these two orders, we calculate the sum of the $\mathcal{V}$-information from X to Y over each token of the sequence under each order and then compute the difference between them as follows:

$$\sum_n I_{\mathcal{V}}(\mathbf{X}_{\mathbf{l}^\tau_{<t}} \to X_{l^\tau_{t+1}}) - \sum_n I_{\mathcal{V}}(\mathbf{X}_{\mathbf{l}^0_{<t}} \to X_{l^0_{t+1}}) \tag{6}$$

$$= \sum_n H_{\mathcal{V}}(X_{l^0_{t+1}}|\mathbf{X}_{\mathbf{l}^0_{<t}}) - \sum_n H_{\mathcal{V}}(X_{l^\tau_{t+1}}|\mathbf{X}_{\mathbf{l}^\tau_{<t}}) \tag{7}$$

This difference, resulting in a value of 0.206, quantifies the increased accessibility of X_{l_t+1} when the context is arranged in the specialized order $\mathbf{l}^\tau$ as opposed to the raster-scan order $\mathbf{l^0}$ within our model family.

C.4 Different Generation Approaches

Relative position encoding. The simplest position encoding assigns an embedding to each absolute patch location l_i $emb^A(l_i)$ (Vaswani et al., 2017). However, we argue that position embedding relative to the current patch may be more effective in learning the generation of the next patch. Note that for the ImageNet experiments, we use absolute positional encodings following RandAR. For the text-to-image setting, we use two types of positional encodings in our approach for our text to image generations: the current patch position is encoded as an absolute position, while the position of the next patch is encoded relative to the current patch. To formulate this, we compute the relative

position embedding for the next patch as: $emb^R(l_{i+1}) = emb^A(S + l_{i+1} - l_i)$, where S is equal to the number of patches in one dimension of the image, A denotes the absolute embedding and R denotes the relative embedding. Note that parameters are not shared between the absolute and relative positional embeddings. In relative positioning, instead of associating a position embedding l_i to each patch in the image, we assign a positional embedding based solely on the distance of a given patch from the current position. This relative position embedding helps the model to learn to generate patches locally as it has direct access to the information on the distance between the next patch and the previous one. This setup enables the transformer to leverage both types of positional information. We use axial positional embeddings (Ho et al., 2020) to model the positional embeddings in both cases. We use the standard image height and width for the absolute part to construct the embedding table, while for the relative part, we use double the height and width to accommodate the relative difference.

Absolute and Relative encodings. We compare the use of absolute and relative encodings. To further investigate the practical implications, we observe how these encodings affect the output by visualizing the order. As illustrated in Figure 8, consecutive tokens in the latent 1D-sequence are positioned more closely when generating the background, which aligns with the measured average distance between consecutive tokens d for both models in Table 8. While the metrics clearly indicate an enhancement in model performance, the visual differences remain subtle and difficult to discern.

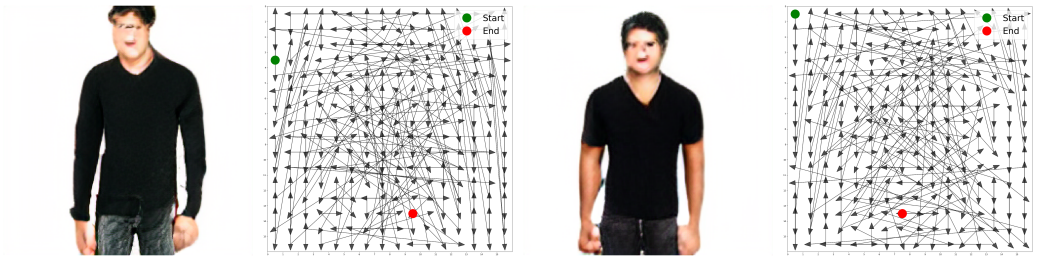

Figure 8: Generation order with absolute and relative positioning encoding. **(Left two)** With absolute encoding the generation is very scattered. **(Right two)** With relative positioning the generation is more localized. The average euclidean distance between the subsequently generated patches in case of absolute encoding is 5.78 whereas in case of relative encoding it is 4.34

Table 8: Generation with absolute and relative positional encoding for the next token.

Method	l_{i+1}	FID (↓)	IS (↑)	KID (↓)	d
Raster	-	4.58	1.106	0.0031	1.83
OAR	Abs.	3.96	1.102	0.0024	5.78
OAR	Rel.	3.02	1.108	0.0019	4.34

C.5 KV Caching

In generative inference, modern LLMs and AR models (Ge et al., 2024; Ramesh et al., 2021) present substantial latency and throughput concerns (Adnan et al., 2024). Key-value (KV) caching significantly reduces the computational cost required by storing the key and value projections of the previously generated tokens to avoid re-computing them later. We should also note that the use of KV caching is mostly enabled by causal masking. In contrast, KV caching may not be implemented for bidirectional attention or other attention strategies (Chang et al., 2022; Li et al., 2024) since the token sequence is not generally fixed.

In the same way, it is not straightforward to use KV caching in our method since the next token position in the sequence is not known in advance. To address this gap and speed up the inference, we remodel the traditional KV caching method (Pope et al., 2023). As described in Section 4.4, our method relies on querying every available location at each AR step and generating a patch at the best one only. To accelerate this inference process, we compute the possible locations in parallel, thereby reducing the overall computation time.

We consider having a temporary secondary cache at each AR step, to store KV representations for all possible locations. Upon selecting the final location, we update the primary cache with the appropriate KV representation from the secondary cache. Once we select the desired location, the secondary KV cache is deleted to free up memory. We provide the broad-level implementation in Algorithm 2.

Algorithm 2 Caching Strategy for text conditioned image processing

Input: Token sequence of length i, list of available locations l
Output: Updated primary cache with selected key-value pair
1: Compute k_i^l and v_i^l for all $l \in$ available_locations in parallel
2: Store all k_i^l and v_i^l in secondary cache
3: Select location l^*
4: Store $k_i^{l^*}$ and $v_i^{l^*}$ in primary cache
5: Delete secondary cache

Impact of Improved KV Caching. We quantify the effect of our two efficiency strategies, namely (i) parallel evaluation of candidate locations and (ii) optimized KV caching, on the time needed to generate a single image (Table 9).

Table 9: Inference Time Comparison Across Different Model Configurations

Model Configuration	Inference Time (sec)
Raster Scan	6.33
Raster Scan (Cached)	2.83
OAR (Naïve)	262.26
OAR (Parallel Evaluation)	73.76
OAR (Parallel Evaluation + Optimized KV Cache)	2.99

We observe that parallelizing the generation process across different locations reduces latency by approximately $3.5\times$. Our optimized KV caching scheme accelerates inference by nearly $90\times$. With these optimizations, our model achieves inference efficiency comparable to traditional AR models that follow a raster-scan approach for a single image. Note that the reported timings are summed over all the steps in model inference and do not include overhead like loading the model, setting up caches etc.

C.6 Effect of Background

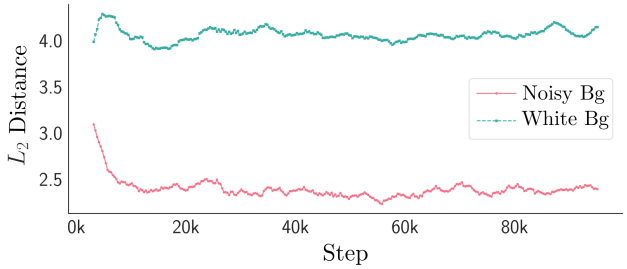

Figure 9: Average distance between generated patches for normal Fashion with white background and our modified version with a noisy background. Due to the different generation orders, the average distance is also different.

All images in the Fashion Products dataset have a large portion covered by white background. While initially, this could be considered an interesting feature to isolate the foreground object, in practice, it introduces biases in the generation process. First, the most frequent patches are the white background

patches. Thus, those patches are the most likely to be generated and, therefore, they will be the first to be generated even though they do not contain any important information about the object of interest (see. Fig.10 (Left two)). To verify that our model can learn orders that prioritize the foreground object, we run an additional training with a modified version of the same dataset, in which the background is filled with random noise instead of white. As the noise varies, the model cannot fit it well, so it would not be as dominant as the white background. In this setting (Fig.10 (Right two)), we observe that the model generates first the object of interest and later the background as expected. Additionally, visually, the generation seems to be improved, as the model can start the generation from the most discriminative part of the image. We do not report FID for this setting, because the Inception network does not produce meaningful features for images with noisy backgrounds, and the resulting FID does not reflect sample quality.

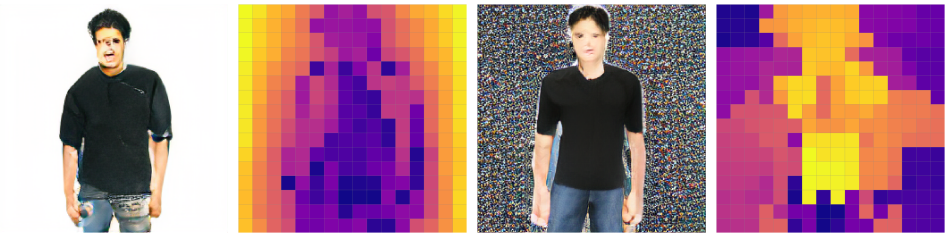

Figure 10: Generation order with different backgrounds. **(Left two)** With an easy background, such as uniform white, the model will start generating from the background. **(Right two)** With a more difficult background, such as random white noise, the model will start generating from the object.

Another interesting variable is the average distance (d) between generated patches. Without the distance regularization of Eq. (8), we checked the evolution of the average distance during training. Results are reported in Fig.9 for the normal dataset and the variation with noisy background. In both cases, the average distance starts from around 3.5. However, when the background is white, the model generates the background first. This increases the average distance to around 4.0. In contrast, when the background is random noise, the model can focus on the foreground first, and the average distance is reduced to around 2.5 over time. This is because when the model focuses on the object, it learns from data that locality is useful for better generation, and therefore, it learns to generate more local patches.

C.7 Distance Regularization

To encourage smoother and more locally coherent generation orders, we add a distance-based regularization term during the selection of the next patch location. At generation step i, the model evaluates all possible patch locations l and selects the next location l_i^* according to

$$l_i^* = \arg\max_l \Big(p_\theta(l, c_{i,l} \mid \mathbf{x}_{<i}, \mathbf{l}_{<i}) - \lambda \, d_\infty(l, l_{i-1}^*) \Big), \tag{8}$$

where:

- $p_\theta(l, c_{i,l} \mid \mathbf{x}_{<i}, \mathbf{l}_{<i})$ is the model's predicted score (or likelihood) for choosing location l at step i, given previously generated patches $\mathbf{x}_{<i}$ and previously selected locations $\mathbf{l}_{<i}$;
- $c_{i,l}$ denotes the predicted content token at location l at step i;
- l_{i-1} is the location chosen at the previous step;
- $d_\infty(l, l_{i-1}^*)$ is the L_∞ distance between location l and the previous selected location l_{i-1};
- λ is a regularization weight controlling the strength of the distance penalty.

This formulation introduces a trade-off between selecting the location with the highest model likelihood and staying spatially close to the previously generated patch. Larger values of λ encourage more local, contiguous generation, whereas smaller values of λ allow the model to prioritize high-likelihood patches even if they are spatially distant.

Table 10 illustrates the impact of different values of λ, which control the trade-off between the patch likelihood and the distance from the previous token, on generation performance. As observed, the

impact of this regularization is minimal. We believe this limited effect is due to the regularization being applied only at generation time, without affecting the learning. We leave the application for similar regularization during training for future work.

Table 10: Generation with different penalty regularization.

λ	FID (↓)	IS (↑)	KID (↓)	d
0.0	3.15	1.106	0.0023	5.34
0.3	3.11	1.106	0.0021	4.65
0.5	3.02	1.108	0.0019	4.34
0.7	3.05	1.106	0.0019	4.09

C.8 Decoding strategies

In our study, we aim to maximize the use of the representation learned by our model to enhance image generation quality. To achieve this, we perform an ablation over decoding strategies. Instead of sampling the location according to Eq. (8), a simple alternative consists of considering the joint representation and then taking the top-k (50% of the codebook size, in our case) elements of all the patches together. We then mask the rest of the elements to ensure they are not selected. Therefore, we see the representation as a joint representation as opposed to sampling the location followed by the content. We observe the FID to slightly drop to 3.04, the IS to be 1.109 and the KID to be 0.0020.

C.9 Impact of Self-Distillation

To evaluate the impact of the self-distillation stage, we compare our final model against a baseline trained under the same framework but without the distillation step. Both models undergo the same total training epochs to ensure a fair comparison, with the only distinction being that our model undergoes self-distillation with extracted specialized orders during the final 150 epochs. After 450 epochs of any-order training, the baseline model achieves an FID of 2.98. In contrast, our model, which undergoes 300 epochs of any-order training followed by 150 epochs of self-distillation, achieves an improved FID of 2.56. These results show that the self-distillation stage substantially reduces FID, highlighting its crucial role in the pipeline.

C.10 Qualitative Samples on Multimodal CelebA-HQ

Figure 11 shows additional text-to-image samples and their generation orders on Multimodal CelebA-HQ.

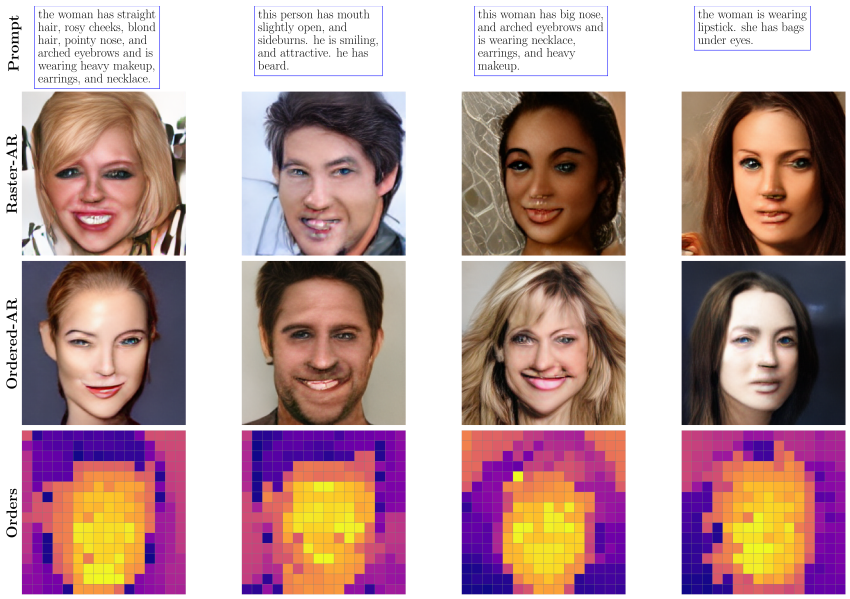

Figure 11: Examples of generation on the CelebA dataset. (Top) Generated images with raster AR mode. (Middle) Generated images with OAR model. (Bottom) Generation order, from yellow to violet. On this dataset our model generates first the salient parts of a face, leaving hair and background at the end. Our model produces images with greater smoothness, rich context and more aligned with the text.