Unlocking Any-Order Generation in Pretrained Autoregressive Image Models

Rishav Pramanik[1] **Marco Pedersoli**[2,3] **Zhaozheng Yin**[1]
[1]Stony Brook University, New York, USA
[2]International Laboratory on Learning Systems (ILLS)
[3]LIVIA, ÉTS Montréal, QC, Canada

Abstract

Autoregressive (AR) image generators are powerful in principle but inflexible in practice: training with a fixed raster-scan order hardcodes a single factorization of the image distribution into the model, leaving it unable to inpaint, outpaint, or edit at inference time. Existing solutions, including the modern discrete-diffusion family, train from scratch on a mixture of token orderings, paying roughly $3\times$ the compute of standard raster-scan training; discrete diffusion further sacrifices compatibility with the AR inference ecosystem (KV caching, vLLM) by construction. We argue that this cost is unnecessary. A pretrained raster-scan AR model has already learned the joint distribution over image tokens; what it lacks is not knowledge but the ability to express that knowledge through non-sequential conditional pathways. We propose a recipe for adapting pretrained raster-scan models into any-order generators using two components: *future-aware positional embeddings* that resolve a target-position identifiability problem at negligible overhead, and an *ordering curriculum* that traces a stable optimization path from the pretrained solution to a good any-order basin. Applied to four base models across two architectures and two scales (LlamaGen-L/XL, RAR-L/XL), the recipe substantially outperforms MaskGIT on ImageNet inpainting (FID 6.21 vs. 10.44) and outpainting, is competitive with MAR-L, trades a 1.2–1.9 increase in class-conditional FID for this flexibility, and costs 13–33% additional compute beyond pretraining, while leaving the standard AR inference stack untouched. Inpainting, outpainting, and editing, our results suggest, need not be added to autoregressive image models. They can be unlocked from the ones that exist. Code is available at `https://github.com/Ordered-AR/unlockedadaptation`.

1 Introduction

Autoregressive (AR) modeling has become a common paradigm for image generation [3, 1, 4, 5], inheriting the scalability and training stability [6, 7, 8, 9] that made it standard in language [10, 11, 12, 13]. By tokenizing images into discrete codes and predicting them sequentially [14, 1, 15, 16], AR models reach generation quality comparable to that of leading diffusion and bi-directional models, while sharing the architecture of their language counterparts. Standard AR image models, however, impose a fixed generation order, typically raster scan, proceeding left to right and top to bottom. This ordering is convenient, but it hardcodes a single factorization of the image distribution into the model, leaving it unable to condition on non-contiguous context at inference time.

Any-order generation removes this restriction. Rather than committing to a single token ordering, an any-order AR model predicts any subset of image tokens conditioned on any other subset. This unlocks a range of capabilities from a single model: by treating observed tokens as context and sampling the remainder, the same weights perform inpainting, outpainting, and region-conditioned

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

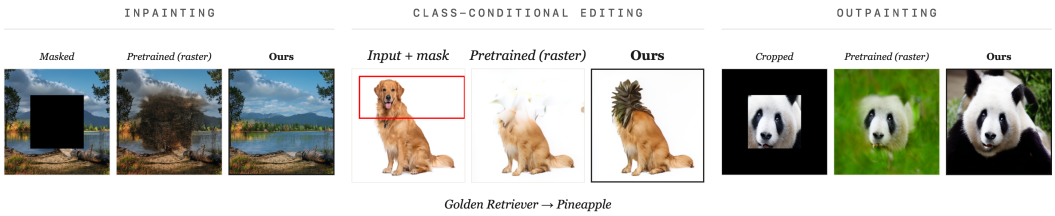

Figure 1: A single autoregressive image model, adapted from a pretrained raster-scan generator, performs inpainting (left), region-conditioned editing (center), and outpainting (right) without task-specific training, sampler modification, or auxiliary networks. The "Pretrained (raster)" column shows the same LlamaGen-L [1] model from which our model is adapted, prompted to perform the same tasks under the natural raster-scan generation procedure. Failure modes in this column reflect the structural incapacity of fixed-order AR models to condition on non-contiguous context, not weak performance at a shared task. After adaptation ("Ours"), the same weights produce coherent results across all three capabilities, varying only in which tokens are observed at inference time. Quantitative comparisons against MaskGIT [2] on inpainting and outpainting are reported in Section 4.

editing without inference-time modification, and admit parallel decoding strategies that predict multiple unobserved tokens jointly rather than strictly one at a time [17, 18]. Diffusion models support analogous tasks, but only through specialized samplers [19, 20] or auxiliary conditioning networks [21, 22] layered onto a fixed generation process. Any-order AR offers them as *zero-shot capabilities* of the native generation procedure: one model, one set of weights, one decoding loop.

The obstacle is cost of training. Existing approaches to any-order image generation train from scratch on a mixture of token orderings, a joint optimization that takes roughly $3\times$ the compute of raster-scan training (e.g., RandAR [18] requires 300 epochs vs. 100 for the corresponding raster-scan LlamaGen-XL [1]). The most common modern approach is *discrete diffusion* [2, 23, 24], which re-frames any-order generation as iterative mask-token denoising under a non-causal transformer. Discrete diffusion reaches competitive quality, but its bidirectional attention and iterative-refinement decoding place it outside the standard AR inference ecosystem: KV caching, speculative decoding, and AR-oriented serving frameworks such as vLLM [25] do not accelerate it. The result is not just a training cost gap but a recurring inference gap: practitioners with existing AR pretraining pipelines and AR serving infrastructure cannot adopt these methods without parallel investment in a separate stack.

We argue that this cost reflects a conceptual conflation rather than a fundamental requirement. *Any-order AR models need not be trained from scratch; they can be adapted from existing pretrained raster-scan models.* A raster-scan AR model has already done the expensive work: it has learned an approximation of the joint distribution over image tokens, encoding texture statistics, spatial coherence, and semantic structure into its representations. The chain rule of probability guarantees that the *true* joint $p(x_1, \ldots, x_n)$ factorizes equivalently under any token ordering, so a model that is a good fit under one factorization is a candidate fit under all of them. We hypothesize that the representations learned under raster-scan training are sufficiently order-agnostic to support any-order generation with light adaptation, because they encode properties of images rather than properties of scan lines. What the model lacks is not distributional knowledge but the ability to express that knowledge through non-sequential conditional pathways, and acquiring that ability, starting from a good initialization, need not be expensive.

To test this hypothesis, we propose a recipe for adapting a pretrained raster-scan AR model into an any-order generator. Two components are jointly necessary. First, we introduce *future-aware positional embeddings* that inject the identity of the next target position into each step of the generation sequence. This addresses an identifiability problem inherent to causal any-order generation: without target-position information, identical context must produce different outputs for different targets, generating conflicting gradient signals that prevent convergence. Our embeddings address this at negligible overhead, preserving the standard single-stream transformer architecture and the inference-time properties it confers. Second, we employ an *ordering curriculum* that gradually interpolates from raster-scan to fully random permutations over the course of adaptation. Without it, sudden exposure to random orderings produces large, high-variance gradients that move the model out of the pretrained basin before order-flexible routing has developed, eroding the representations we

aim to reuse. The curriculum traces a stable optimization path from the pretrained solution to a good any-order basin. Both components are necessary: ablating either causes training to diverge or generation quality to drop noticeably (Section 4).

The resulting model reaches any-order generation at roughly $2.4\times$ lower end-to-end cost than training from scratch, requiring only 25% additional computation beyond the original raster-scan pretraining budget on LlamaGen-XL (13–33% across base models); for practitioners who already have a pretrained AR model, the marginal adaptation cost is over an order of magnitude lower ($12\times$) than training any-order from scratch. On LlamaGen-XL, FID moves from 2.63 to 3.91 while Inception Score, Precision, and Recall remain close to the pretrained values (252.94, 0.81, 0.53), a measured quality-flexibility trade-off rather than a lossless conversion. We observe a similar pattern across LlamaGen-L and RAR-L/XL (Section 4). The adapted model also gains the any-order capabilities promised by the construction (inpainting, outpainting, and region-conditioned editing), where it substantially outperforms MaskGIT [2] on ImageNet inpainting (FID 6.21 vs. 10.44) and outpainting (FID 5.84 vs. 12.17) and is competitive with MAR [3], despite never having been trained on either task (Section 4). Because the recipe leaves the causal transformer architecture untouched, the adapted model retains KV caching and parallel decoding and can be served with vLLM [25] through a custom model registration, whereas discrete-diffusion any-order approaches cannot reuse the KV cache, by construction. Flexible image generation (inpainting, outpainting, editing) need not be added to autoregressive models. It is, in our results, already accessible from the ones that exist.

2 Related Work

Autoregressive image generation. Autoregressive image models, long standard in language modeling, have recently reached generation quality on par with leading diffusion methods on standard benchmarks [3, 1, 4, 26]. They tokenize images via a quantized autoencoder [14, 1, 4, 5] and predict discrete tokens with a transformer, inheriting the inference ecosystem developed for LLMs (KV caching, paged attention, tensor/pipeline parallelism) [25] and integrating naturally into unified multimodal stacks [16, 27]. Standard AR image models commit to a fixed raster-scan order; this work relaxes that constraint while preserving the inference-time properties that make AR attractive.

Any-order generation. Permutation language modeling [28] showed that a single causal model can generate under arbitrary orderings if a target-position identifiability problem is resolved, originally via two-stream attention. Image-domain follow-ups [18, 29, 30] apply the same recipe to discrete image tokens but pay similar costs: any-order capability obtained from scratch, with architectural overhead or training budgets several times that of standard raster-scan AR. MAR [3] instead predicts continuous tokens with a diffusion head, also from scratch. The current dominant alternative is *discrete diffusion* [2, 23, 31], which casts any-order generation as iterative mask-token denoising under bidirectional attention but, by construction, sits outside the AR inference ecosystem. We instead retain causal AR, address identifiability with a lightweight additive embedding, and obtain any-order capability through adaptation rather than from-scratch training.

Adaptation of pretrained models. Adapting pretrained models to new objectives is standard practice in language and vision [32, 33, 34, 35]. Most prior work targets new *tasks* or *domains*; we instead target a new *factorization* of the same data distribution. Task-specific methods such as ControlNet [36] are complementary: they add a branch for one fixed control signal, whereas we support arbitrary observed-token subsets. Whether representations learned under one factorization transfer to others is an empirical question; our results suggest that they do, with light adaptation.

3 Method

We consider an autoregressive image model that predicts a sequence of discrete tokens $x = (x_1, \ldots, x_n)$ produced by a tokenizer applied to an image. A standard AR model factorizes the joint under a fixed raster-scan order $\sigma_{\text{raster}} = (1, \ldots, n)$. An any-order model relaxes this by allowing σ to vary, and is trained to maximize the expected log-likelihood under a distribution over permutations:

$$\mathcal{L}_{\text{any-order}}(\theta) = \mathbb{E}_{\sigma \sim P}\left[\sum_{t=1}^{n} \log p_\theta\big(x_{\sigma(t)} \,\big|\, x_{\sigma(<t)}\big)\right], \tag{1}$$

with $P = \mathrm{Uniform}(S_n)$ for full any-order training. By the chain rule of probability, every choice of σ is a valid factorization of the same joint. The question we address is whether a model pretrained under raster scan can be adapted, by continued training under Equation 1, into one that supports the full family of factorizations. We argue that this is possible, but only if two distinct issues are addressed.

The identifiability problem. Under raster-scan, the prediction target at step t is implicitly $\sigma(t+1) = t+1$ and the model never needs to be told *which* position to predict. Under varying σ, that contract breaks: the same observed context $x_{\sigma(<t)}$ corresponds to different prediction targets across permutations. Concretely, predicting a sky token given the rest of an image is a different problem from predicting a face token given the same context: the visible information is identical, but the conditional distributions are not. A model that attempts to fit $p(x_j \mid x_S)$ and $p(x_k \mid x_S)$ for $j \neq k$ from the same input representation therefore receives conflicting gradients, and the objective in Equation 1 becomes ill-posed without target-position information. XLNet [28] resolved this in language with a two-stream attention mechanism that maintains a separate query stream for target-position information; this restores identifiability but doubles per-layer attention compute and breaks compatibility with KV caching. We propose an alternative based on input-level conditioning (Section 3.1).

The optimization problem from a pretrained initialization. Even with identifiability resolved, naive any-order training from the pretrained parameters θ^* fails. At θ^*, the per-permutation loss $\mathcal{L}_\sigma$ is small for $\sigma \approx \sigma_{\mathrm{raster}}$ but large for random σ, and an expectation over $\mathrm{Uniform}(S_n)$ is therefore dominated by high-loss random permutations. The resulting gradients are large in magnitude and inconsistent in direction across permutations; applied at θ^*, they push the model out of the pretrained basin before order-flexible routing has developed, a form of catastrophic forgetting that erodes the very representations we aim to reuse. We address this with an ordering curriculum (Section 3.2) that interpolates from raster-scan to uniform random over the course of adaptation, tracing a stable optimization path from θ^* to a good any-order basin. Note that this is distinct from simply lowering the learning rate: a smaller step size leaves the gradient *direction* unchanged and continues to point away from the pretrained basin. The curriculum instead modifies the *composition* of the loss, weighting raster-like permutations heavily early on so that the gradient itself is dominated by directions consistent with the pretrained solution. Appendix A gives the full gradient-magnitude and gradient-variance argument and frames it as a continuation method. The two mechanisms address orthogonal aspects of the problem and are jointly necessary: identifiability makes the loss well-posed; the curriculum makes the well-posed loss reachable from the pretrained initialization. Section 4.4 confirms both halves empirically.

3.1 Future-Aware Positional Embeddings

We introduce a learned embedding table $E \in \mathbb{R}^{n \times d}$, where d is the model's hidden dimension and row E_j encodes the identity of position j as a target. Under permutation σ, the input representation at sequence position t is augmented with the embedding of the *next* target position:

$$\tilde{h}_t = h_t + E_{\sigma(t+1)}, \tag{2}$$

where $h_t = e(x_{\sigma(t)})$ is the standard input embedding of the token at sequence position t (for RAR, including its absolute position embedding at $\sigma(t)$). The final token has no successor; we use a learned null embedding $E_\varnothing$. The embedding E is shared across sequence positions and permutations: the same row E_j is added at any sequence position whose successor under σ is the spatial position j. The addition is at the input layer only; we considered AdaLN-style injection at every layer [37] but found a single input-level addition sufficient, consistent with the view that the embedding's role is to disambiguate the prediction target rather than to participate in feature computation throughout the network.

Position handling under permutation. The two base-model families we adapt encode position differently (LlamaGen via rotary position embeddings (RoPE) [38] and RAR via additive learned absolute position tables at the input layer), but the principle for any-order generation is the same in both: at sequence step t, the position used for rotation (LlamaGen) or table look-up (RAR) is the spatial index $\sigma(t)$, not the sequence step. With this choice, attention or input addition between tokens at sequence steps t and t' depends only on their spatial positions $\sigma(t)$ and $\sigma(t')$, identical to what the

model would have computed under raster scan with those tokens at the same spatial locations. The model thus inherits the spatial-relational priors learned during pretraining unchanged; only the order in which sequence steps consume those positions varies.

Compared to two-stream attention [28], the additive embedding adds only $(n+1) \cdot d$ parameters and an elementwise sum, leaves the standard single-stream transformer architecture untouched, and preserves KV caching and parallel decoding.

3.2 Ordering Curriculum

We sample training permutations from a mixture parametrized by a curriculum coefficient $\alpha \in [0, 1]$:

$$\sigma \sim P_\alpha = (1-\alpha) \cdot \delta_{\sigma_{\text{raster}}} + \alpha \cdot \text{Uniform}(S_n). \tag{3}$$

At $\alpha = 0$, adaptation reduces to continued raster-scan training; at $\alpha = 1$, every example is trained on a uniformly random permutation. We use a linear schedule $\alpha(s) = \min(s/S_{\text{ramp}}, 1)$, holding $\alpha = 1$ after step S_{ramp}. Within a minibatch, each example draws an independent permutation, smoothly interpolating between the raster-scan and any-order objectives as α increases.

3.3 Training Objective

The adaptation loss combines the future-aware embedding with the curriculum sampler:

$$\mathcal{L}(\theta) = \mathbb{E}_{\sigma \sim P_{\alpha(s)}} \left[\sum_{t=1}^{n} -\log p_\theta\big(x_{\sigma(t)} \,\big|\, x_{\sigma(<t)},\, E_{\sigma(t+1)}\big) \right]. \tag{4}$$

Apart from the curriculum sampler and the additive embedding, the training pipeline is identical to continued raster-scan AR training. We initialize from the pretrained checkpoint and adapt for T_{ft} epochs (set per base model so adaptation cost is a small fraction of pretraining). We use the same optimizer and weight decay as pretraining; S_{ramp} is tuned on a held-out subset. The future-aware embedding table E is zero-initialized (for RAR, its existing target-aware positional embedding is loaded from the checkpoint) and trained at $10\times$ the base learning rate (Section 4.4); per-model values are in Appendix B.

3.4 Inference

At inference, the user (or upstream task) specifies a generation order σ: raster scan recovers the pretrained model's behavior; uniform random shuffles the order; task-induced placements put observed tokens (an inpainting mask, an outpainting boundary, a partial edit) first in σ and sample the rest in any chosen order. Generation proceeds sequentially under σ with the future-aware embedding $E_{\sigma(t+1)}$ added to each input. Because the architecture is a standard single-stream transformer, KV caching applies unchanged, supporting inpainting, outpainting, and region-conditioned editing as direct instances of conditional generation without task-specific training, sampler modification, or auxiliary networks (Section 4).

4 Results

We evaluate our adaptation recipe along four axes: (i) generation quality relative to standard generative baselines and the pretrained raster-scan models we adapt (Section 4.2); (ii) the any-order capabilities the unlocked model acquires, demonstrated on inpainting, outpainting, and region completion (Section 4.3); (iii) the necessity of each recipe component, validating the diagnoses in Section 3 (Section 4.4); and (iv) training and inference cost relative to existing any-order approaches (Section 4.5).

4.1 Experimental Setup

Datasets. All experiments use ImageNet-1k (ILSVRC-2012) [39], on which all base models and baselines were pretrained. Adaptation uses the 1.28M-image training split, pre-tokenized with each model's VQ tokenizer. Class-conditional metrics are computed against the standard ImageNet 256×256 reference statistics [40], inpainting and outpainting are evaluated on the 50,000-image

validation split, and region completion on $5{,}000$ held-out images, with masks shared across methods. We use no other datasets.

Base models. We apply our recipe to two families of class-conditional autoregressive image generators trained on ImageNet: LlamaGen [1] (sizes L and XL) and RAR [5] (sizes L and XL). For each base model we start from the publicly released pretrained checkpoint and apply the recipe described in Section 3. Image tokens come from the corresponding pretrained VQ tokenizer; we do not modify the tokenizer. LlamaGen-XL generates at 384×384 following its released configuration and is resized to 256×256 for evaluation.

Adaptation. We adapt each pretrained checkpoint for T_{ft} epochs (set per base model; see Appendix B) with the curriculum coefficient α scheduled linearly from 0 to 1 over the first S_{ramp} epochs and held at 1 for the remainder. Future-aware embeddings are zero-initialized (loaded from the checkpoint for RAR) and trained at $10\times$ the base learning rate; we ablate this choice in Section 4.4. All other hyperparameters match the corresponding pretraining configuration.

Evaluation. We report Fréchet Inception Distance (FID) [41], Inception Score (IS) [42], and Precision/Recall on $50{,}000$ generated samples following the standard ImageNet protocol. All samples use the same number of decoding steps as the corresponding pretrained model and the published sampling configuration unless otherwise specified.

4.2 Class-Conditional Image Generation

Table 1 reports class-conditional generation quality on ImageNet 256×256, alongside representative diffusion and AR baselines. Adaptation trades some raster-scan quality for any-order capability:

Table 1: Class-conditional generation on ImageNet 256×256. Our adapted models (rows marked **Ours**) trade a 1.17–1.93 FID increase relative to the pretrained raster-scan baselines they are adapted from for full any-order capability (Section 4.3). *Reported by the original paper.

Type	Model	#Para.	FID↓	IS↑	Precision↑	Recall↑	Steps
Diffusion	ADM [40]	554M	4.59	186.70	0.82	0.523	250
	LDM-4 [43]	400M	3.60	247.7	–	–	250
	DiT-XL [37]	675M	2.27	278.2	0.83	0.57	250
Bi-directional AR	MaskGIT-re [2]	227M	6.80	214.0	0.82	0.51	8
	TiTok-S-128 [44]	287M	1.97	281.8	–	–	64
Causal AR	VAR [4]	600M	2.57	302.6	0.83	0.56	10
	VQGAN-re [14]	1.4B	5.20	280.3	–	–	256
	RAR-B [5]	261M	1.95	290.5	0.82	0.58	256
	RAR-L [5]	461M	1.70	299.5	0.81	0.60	256
	RAR-XL [5]	955M	1.50	306.9	0.80	0.62	256
	LlamaGen-L [45]	343M	3.07	256.06	0.83	0.52	256
	LlamaGen-XL* [45]	775M	2.63	244.08	0.80	0.57	576
	RandAR-L [18]	343M	2.55	288.5	0.81	0.58	88
	RandAR-XL [18]	775M	2.22	314.21	0.80	0.60	256
Ours *(Causal AR)*	LlamaGen-L + adaptation	343M	4.24	231.62	0.82	0.52	256
	LlamaGen-XL + adaptation	775M	3.91	252.94	0.81	0.53	576
	RAR-L + adaptation	461M	3.63	283.13	0.77	0.60	256
	RAR-XL + adaptation	955M	2.68	285.16	0.80	0.58	256

across four base models, FID increases by between 1.17 and 1.93 relative to the corresponding pretrained checkpoint, while IS, Precision, and Recall remain comparable. We do not claim this cost is fundamental or monotonic with scale. The absolute gap is somewhat larger on RAR (which sits closer to the FID floor, where the same nominal degradation is more visible) than on LlamaGen, but the recipe transfers without modification across both architectures and both scales. This reduction in raster-scan quality (see Fig 2) buys the full any-order capability suite (Section 4.3) at $\approx 1.25\times$ the original pretraining budget on LlamaGen-XL, against the $\approx 3\times$ required by from-scratch any-order training (Section 4.5).



Figure 2: Class-conditional generation after adaptation, further accelerated by parallel decoding. **Top:** pretrained LlamaGen-L (raster). **Middle:** our adapted model with sequential decoding. **Bottom:** our adapted model with parallel decoding (64 steps, $4\times$ fewer forward passes).

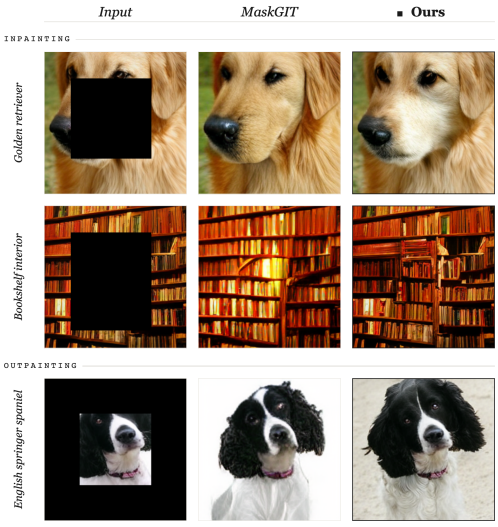

Figure 3: Qualitative comparison on inpainting (top) and outpainting (bottom) against MaskGIT.



Figure 4: Training loss under our full recipe and the future-aware embedding ablation. **Without future-aware embeddings** (red), the loss fails to descend and instead climbs as the curriculum ramps in random orderings, reflecting the identifiability conflict diagnosed in Section 3. The full recipe (blue) descends smoothly throughout.

Value of pretraining. At matched total epochs, training the same architecture and curriculum from random initialization is worse than adapting the pretrained checkpoint (Table 2). This control matches compute but not schedule, since only the pretrained pathway begins with raster-only training.

Table 2: Value of pretraining: FID at matched total epochs with the same architecture and curriculum.

Model	Total epochs	Pretrained + adapted ↓	Random init ↓
LlamaGen-L	400 (300+100)	**4.24**	5.60
LlamaGen-XL	125 (100+25)	**3.91**	4.81

4.3 Any-Order Capabilities: Inpainting, Outpainting, and Editing

A model that can generate in any order can perform inpainting and outpainting as direct instances of conditional generation under task-induced orderings (Section 3.4). We evaluate both tasks against

MaskGIT [2], with its original cosine schedule and with the Halton schedule [46], and against MAR-L [3], using released ImageNet checkpoints, the same evaluation protocol, and the same masks across methods.

Table 3: Inpainting and outpainting on ImageNet 256×256, evaluated on $50{,}000$ samples with TF-Inception. Inpainting uses center masks; outpainting extends a centered 128×128 crop to full resolution. Lower is better for FID; higher for IS, Precision, Recall.

	Inpainting				Outpainting			
Method	FID ↓	IS ↑	Prec. ↑	Rec. ↑	FID ↓	IS ↑	Prec. ↑	Rec. ↑
MaskGIT (cosine)	10.44	116.2	0.680	0.575	12.17	107.0	0.664	0.631
MaskGIT (Halton)	7.82	123.5	0.581	0.575	7.91	119.1	0.691	0.612
MAR-L	6.63	134.2	0.696	**0.598**	6.86	127.2	**0.736**	**0.649**
Ours	**6.21**	**157.0**	**0.710**	**0.598**	**5.84**	**159.4**	0.720	0.637

Our adapted model outperforms both MaskGIT schedules (Table 3), although Halton narrows the gap, so part of the original margin was scheduler-dependent. Against MAR-L, our FID is lower on both tasks while MAR-L has higher outpainting Precision and Recall; without run-level uncertainty, we regard the two as competitive. The qualitative comparison in Figure 3 bears this out at the per-image level: under inpainting, our method recovers sharper structure and more coherent global lighting than MaskGIT; under outpainting, it synthesises a plausible scene around the cropped subject while preserving identity and palette.

Region-conditioned editing. On $5{,}000$ held-out ImageNet images, we regenerate a random contiguous region covering 30–50% of the image from the rest and measure LPIPS to the original; we also report PSNR and SSIM over the generated region of the center-mask inpainting examples. Our model is best on all three (Table 4). Since several completions can be valid, these reference-based metrics complement FID, and they do not measure instruction following or identity preservation, which remain qualitative.

4.4 Ablations

The motivation in Section 3 makes two falsifiable predictions: future-aware positional embeddings are necessary to make the any-order objective well-posed, and the ordering curriculum is necessary to make it reachable from a pretrained initialization. We test each prediction directly.

Future-aware embeddings are necessary. Removing future-aware embeddings (Figure 4) prevents the loss from descending below a substantial floor: as the curriculum ramps in random orderings, the same observed context must explain progressively more diverse target distributions, and the loss *increases* rather than decreases. This is the gradient-conflict failure mode predicted by the identifiability argument in Section 3, observed empirically. The corresponding generation quality is poor across all evaluation orders.

The curriculum is necessary. With $\alpha = 1$ from step zero (i.e., applying the well-posed any-order objective to the pretrained checkpoint without any schedule), at convergence the no-curriculum model trails our full recipe across all fidelity metrics on LlamaGen-L (Table 5), while Recall is essentially unchanged, a fidelity collapse rather than a coverage collapse. The pretrained representations were eroded before order-flexible routing could form.

Embedding learning rate. The future-aware embedding table E is zero-initialized, which means it must develop useful values from scratch even as the curriculum increases the model's reliance on it. We find empirically that scaling the learning rate of E by $10\times$ relative to the rest of the network is necessary for stable convergence: at the base learning rate, the embedding develops too slowly to keep pace with the curriculum, and final any-order quality suffers. We use the $10\times$ multiplier in all main results.

Alternative mitigation: teacher regularization. Adding a KL term ($\lambda = 1.0$) toward the frozen raster-scan teacher on raster-ordered examples raises LlamaGen-L FID from 4.24 to 9.62, suggesting that matching raster-specific predictions conflicts with learning arbitrary-order conditionals.

Table 4: Reference-based completion metrics. LPIPS: random region covering 30–50% of 5,000 held-out images. PSNR/SSIM (mean ± std): generated region of the 50,000 center-mask inpainting examples.

Method	LPIPS ↓	PSNR ↑	SSIM ↑
MaskGIT (cosine)	0.234	19.20 ± 3.60	0.484 ± 0.21
MaskGIT (Halton)	0.196	18.20 ± 3.50	0.452 ± 0.23
MAR-L	0.174	20.15 ± 4.60	0.550 ± 0.20
Ours	**0.125**	$\mathbf{24.65 \pm 3.34}$	$\mathbf{0.682 \pm 0.17}$

Table 5: Effect of the ordering curriculum on LlamaGen-L. Adapting with $\alpha = 1$ from step zero (no curriculum) leaves the model in a partially adapted state across all metrics; the linear curriculum used in our main results recovers the gap.

Variant	FID ↓	IS ↑	Prec. ↑	Rec. ↑
No curriculum ($\alpha = 1$ from step 0)	5.86	191.27	0.78	0.51
Ours (linear curriculum)	**4.24**	**231.62**	**0.83**	**0.52**

4.5 Training and Inference Efficiency

The central practical claim of this paper is that any-order capability can be obtained at a small fraction of the cost of training such a model from scratch. Table 6 reports total training compute cost, expressed as a multiple of the corresponding raster-scan pretraining budget, alongside compatibility with the standard AR inference ecosystem.

Table 6: Training and inference compatibility relative to standard AR pretraining. Our adaptation recipe achieves any-order generation at $\approx 1.25\times$ the cost of standard pretraining (LlamaGen-XL), while preserving compatibility with the AR inference ecosystem.

Method	Train cost	Any-order	KV cache	vLLM
LlamaGen pretraining [1] (raster only)	1×	✗	✓	✓
RandAR [18] (from scratch)	3×	✓	✓	✗
RAR [5] (from scratch)	4×	✗	✓	✓
Discrete diffusion [2]	3×	✓	✗	✗
Ours (LlamaGen-XL + adaptation)	**1.25×**	✓	✓	✓

Three observations follow. First, the cost of unlocking any-order generation is roughly $1.25\times$ the original pretraining run on LlamaGen-XL (1.13–$1.33\times$ across base models, lowest on RAR-XL, where adaptation amortizes over a longer pretraining run), compared to $\approx 3\times$ for from-scratch any-order training, with the gap widening as the base model grows. The future-aware embedding adds $(n+1) \cdot d$ parameters, 0.74M (0.095%) for LlamaGen-XL. Second, our model retains compatibility with the AR inference ecosystem (KV caching, tensor and pipeline parallelism, and vLLM [25] serving through a custom model registration that supplies spatial position indices and the future-aware embedding) because the recipe leaves the standard causal single-stream transformer untouched. Discrete-diffusion any-order approaches [2, 23] sacrifice this compatibility by construction: their non-causal attention re-attends to all unmasked positions at every refinement step, incompatible with the per-token cache reuse that powers AR inference frameworks. Third, the model admits *parallel decoding* as a direct consequence of any-order training: tokens whose conditionals are near-independent given observed context can be sampled jointly within a single forward pass. On LlamaGen-XL, $9\times$ fewer forward passes cost 1.12 FID (Table 7); Figure 2 (bottom row) shows LlamaGen-L at 64 steps.

Table 7: Parallel decoding on LlamaGen-XL: generation steps vs. class-conditional FID.

Generation steps	576	192	128	64
FID ↓	3.91	4.12	4.81	5.03

At maximum native-setting throughput on one RTX PRO 6000, our model takes 35.6 ms per image (batch 64), against 56.3 ms for MAR-L and 23–24 ms for MaskGIT, whose 12-step decoders are faster but reach worse inpainting FID; this is not a matched-FID comparison.

5 Conclusion

We presented a recipe for converting a pretrained raster-scan autoregressive image generator into an any-order generator through targeted adaptation. The recipe combines a future-aware positional embedding that resolves the identifiability problem inherent to causal any-order generation, and an ordering curriculum that traces a stable optimization path from the pretrained solution to a good any-order basin. Across four base models spanning two architectures and two scales (LlamaGen-L/XL, RAR-L/XL), the adapted models trade a 1.17–1.93 increase in class-conditional FID relative to the pretrained checkpoints for any-order capability, outperform MaskGIT on inpainting (FID 6.21 vs. 10.44) and outpainting (FID 5.84 vs. 12.17) and are competitive with MAR-L despite never being trained on either task, and remain compatible with the standard AR inference ecosystem (KV caching, parallel decoding, vLLM via a custom model). For LlamaGen-XL, the marginal cost of adaptation is approximately 25% of the original pretraining budget, against the $\approx 3\times$ required by from-scratch any-order training.

The chain rule of probability guarantees that any model that fits the joint distribution under raster-scan order is, in principle, a fit under every other ordering as well. Our results suggest that this principle is also empirically actionable: the representations a pretrained raster-scan model has already learned are not specific to scan order, and unlocking access to non-sequential conditional pathways is a much smaller intervention than learning the joint from scratch. We do not claim that causal AR should replace diffusion or flow matching; our contribution is reusing existing causal checkpoints.

Limitations. Our experiments are restricted to class-conditional image generation on ImageNet at 256×256; whether the recipe extends to text-to-image AR models or higher resolutions remains to be tested. We adapt token-based AR models specifically, and our diagnosis of the identifiability problem applies to any causal any-order generator; non-token-based variants such as MAR (which predicts continuous tokens via a diffusion head) lie outside the scope of models we adapt. Our editing evaluation covers region completion only. Finally, our random-initialization control (Section 4.2) matches compute but not schedule.

Acknowledgments and Disclosure of Funding

Rishav Pramanik and Zhaozheng Yin have been supported by NSF grants CMMI-2246673 and CAIF-2331769, and SUNY AI Platform Award. Marco Pedersoli was supported by the Natural Sciences and Engineering Research Council of Canada.

References

[1] 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.

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

[3] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37:56424–56445, 2024.

[4] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

[5] Qihang Yu, Ju He, Xueqing Deng, Xiaohui Shen, and Liang-Chieh Chen. Randomized autoregressive visual generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18431–18441, 2025.

[6] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[7] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

[8] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, 2022.

[9] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR, 2022.

[10] Nicholas Carlini, Newton Cheng, Keane Lucas, Michael Moore, Milad Nasr, Vinay Prabhushankar, Winnie Xiao, Hakeem Angulu, Evyatar Ben Asher, Jackie Bow, Keir Bradwell, Ben Buchanan, David Forsythe, Daniel Freeman, Alex Gaynor, Xinyang Ge, Logan Graham, Kyla Guru, Hasnain Lakhani, Matt McNiece, Mojtaba Mehrara, Renee Nichol, Adnan Pirzada, Sophia Porter, Andreas Terzis, and Kevin Troy. Assessing claude mythos preview's cybersecurity capabilities, March 2026. Frontier Red Team Report.

[11] OpenAI. Introducing GPT-5.3-Codex, February 2026.

[12] Qwen Team. Qwen3.6-Plus: Towards real world agents, April 2026.

[13] Gemini Team, Google. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

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

[15] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

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

[17] Rishav Pramanik, Amin Sghaier, Masih Aminbeidokhti, Juan A. Rodriguez, Antoine Poupon, David Vazquez, Christopher Pal, Zhaozheng Yin, and Marco Pedersoli. Distilling specialized orders for visual generation, 2026.

[18] 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 *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 45–55, 2025.

[19] Shu Zhang, Xinyi Yang, Yihao Feng, Can Qin, Chia-Chih Chen, Ning Yu, Zeyuan Chen, Huan Wang, Silvio Savarese, Stefano Ermon, et al. Hive: Harnessing human feedback for instructional visual editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9026–9036, 2024.

[20] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11461–11471, 2022.

[21] Elior Benarous, Yilun Du, and Heng Yang. Spie: Semantic and structural post-training of image editing diffusion models with ai feedback. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6395–6407, 2025.

[22] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022.

[23] Mark Weber, Lijun Yu, Qihang Yu, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. Maskbit: Embedding-free image generation via bit tokens. *Transactions on Machine Learning Research*, 2024. Featured Certification, Reproducibility Certification.

[24] Hu Yu, Hao Luo, Hangjie Yuan, Yu Rong, and Feng Zhao. Frequency autoregressive image generation with continuous tokens. *arXiv preprint arXiv:2503.05305*, 2025.

[25] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626, 2023.

[26] Sucheng Ren, Qihang Yu, Ju He, Xiaohui Shen, Alan Yuille, and Liang-Chieh Chen. Beyond next-token: Next-x prediction for autoregressive visual generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15781–15791, 2025.

[27] Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

[28] 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.

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

[30] Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. In *International Conference on Learning Representations*, 2022.

[31] Huiwen Chang, Han Zhang, Jarred Barber, Aaron Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Patrick Murphy, William T Freeman, Michael Rubinstein, et al. Muse: Text-to-image generation via masked generative transformers. In *International Conference on Machine Learning*, pages 4055–4075. PMLR, 2023.

[32] Zixuan Ke, Haowei Lin, Yijia Shao, Hu Xu, Lei Shu, and Bing Liu. Continual training of language models for few-shot learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10205–10216, 2022.

[33] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[34] 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 *International Conference on Learning Representations*, 2022.

[35] Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Transactions on Machine Learning Research*, 2024.

[36] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3813–3824. IEEE, 2023.

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

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

[39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

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

[41] Tuomas Kynkäänniemi, Tero Karras, Miika Aittala, Timo Aila, and Jaakko Lehtinen. The role of ImageNet classes in Fréchet Inception Distance. In *The Eleventh International Conference on Learning Representations*, 2023.

[42] Shane Barratt and Rishi Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

[43] 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*, pages 10684–10695, 2022.

[44] 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*, 2024.

[45] 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.

[46] Victor Besnier, Mickael Chen, David Hurych, Eduardo Valle, and Matthieu Cord. Halton scheduler for masked generative image transformer. In *International Conference on Learning Representations*, volume 2025, pages 83027–83047, 2025.

A Extended Motivation

This appendix expands the gradient-magnitude and gradient-variance argument summarized in Section 3, and frames the ordering curriculum as a continuation method.

Per-permutation loss. Recall the any-order objective, Equation 1. With future-aware embeddings making the per-step target identifiable, the per-permutation loss at parameters θ is

$$\mathcal{L}_\sigma(\theta) = -\sum_{t=1}^{n} \log p_\theta\big(x_{\sigma(t)} \,\big|\, x_{\sigma(<t)},\, \sigma(t+1)\big). \tag{5}$$

At the pretrained initialization θ^*, this loss takes very different values depending on σ. For σ near raster scan, $\mathcal{L}_\sigma(\theta^*)$ is small: the model is already a good fit. For random σ, $\mathcal{L}_\sigma(\theta^*)$ is large: the model has never been trained to predict in such orders.

Gradient magnitude. The gradient norms $\|\nabla\mathcal{L}_\sigma(\theta^*)\|$ inherit this imbalance: small for raster-like permutations, large for random ones. The expected gradient under $\mathrm{Uniform}(S_n)$ is therefore dominated by high-loss permutations. Even though most $\sigma \neq \sigma_{\mathrm{raster}}$ are unfamiliar to the model, they collectively contribute the bulk of the gradient signal at θ^*. Early updates therefore optimize aggressively for random-permutation loss, and in doing so they rapidly modify the parameters that encode the raster-scan distribution, the very representations we wish to preserve.

Gradient variance. Different random permutations σ_1, σ_2 induce conditionals with different dependency structures and therefore prescribe different parameter updates. The directions $\nabla \mathcal{L}_{\sigma_1}(\theta^*)$ and $\nabla \mathcal{L}_{\sigma_2}(\theta^*)$ can disagree substantially. Stochastic gradients computed over batches of random permutations therefore have high variance, causing optimization to oscillate rather than converge and, combined with the magnitude problem, to oscillate in directions that destabilize the pretrained basin.

The combined effect is catastrophic forgetting: the network's order-flexible routing has not yet developed when its raster-scan competence is being eroded, leaving the model worse on raster scan and not yet good on anything else.

Curriculum as a continuation method. The curriculum addresses both failure modes. At small α in Equation 3, the gradient is dominated by raster-scan terms whose magnitude is small at θ^*, and the model stays close to the pretrained basin. The small contribution from random permutations is enough to begin shaping order-flexible routing without destabilizing the existing representations. As α increases, the model has already partially adapted, so the random-permutation loss at the current parameters is no longer catastrophically large; magnitude and variance across random permutations both decrease. This is a *continuation method*: rather than leaping from θ^* to a good any-order solution in a single optimization, we trace a path through parameter space along which each successive objective is a small perturbation of the last. At every intermediate α, the previous solution is a good initialization for the current objective, and the model's competence on prior orderings is preserved as a foundation for new ones.

Design-choice asides. We considered injecting the future-aware embedding $E_{\sigma(t+1)}$ at every transformer layer (analogous to AdaLN-style conditioning [37]), but found a single input-layer addition sufficient, consistent with the view that the embedding's role is to disambiguate the prediction target rather than to participate in feature computation throughout the network.

B Hyperparameters

Table 8 reports the full set of hyperparameters used to adapt each pretrained checkpoint with our recipe. All models are adapted on ImageNet from the publicly released raster-scan checkpoints at their released resolutions. The adaptation budget T_{ft} is set per base model to amortize against the corresponding pretraining cost (see Section 4.5). The curriculum coefficient α ramps linearly from 0 to 1 over the first S_{ramp} epochs and remains at 1 for the remainder of adaptation; we report S_{ramp} per model below.

The remaining configuration (architecture, tokenizer, conditioning) matches the corresponding pretraining setup of each base model and is unchanged by adaptation. For LlamaGen, the future-aware embedding adds $(n+1) \cdot d$ parameters, where n is the sequence length, d the hidden dimension, and the extra row is the null embedding (0.26M for LlamaGen-L, 0.74M for LlamaGen-XL). RAR already contains a target-aware positional embedding, which we load from its checkpoint.

Global batch size is not fixed across runs: it depended on the GPU configuration of the node we were allocated, with the per-run setting chosen to maximize GPU utilization on that allocation. Total effective batch size was held within the range used by each base model's pretraining configuration via gradient accumulation when needed.

C Training and Compute Details

Hardware. All adaptation experiments were run on internal clusters with two node types:

- **H200 nodes:** 8×NVIDIA H200 (144 GB HBM each), 192 CPU cores per node.
- **RTX PRO 6000 nodes:** 4×NVIDIA RTX PRO 6000 Blackwell Server Edition (96 GB each), 64 CPU cores per node.

Smaller base models (LlamaGen-L, RAR-L) were adapted on the RTX PRO 6000 cluster; larger models (LlamaGen-XL, RAR-XL) used H200 nodes to accommodate the larger sequence length and per-step memory. LlamaGen-XL adaptation took ≈ 64 H200-GPU-hours (one $8\times$H200 node for ≈ 8 hours); we did not log per-run GPU-hours for the other models.

Table 8: Adaptation hyperparameters by base model. Notation matches Section 3.

	LlamaGen-L	LlamaGen-XL	RAR-L	RAR-XL
Optimization				
Optimizer	AdamW	AdamW	AdamW	AdamW
(β_1, β_2)	$(0.9, 0.95)$	$(0.9, 0.95)$	$(0.9, 0.95)$	$(0.9, 0.95)$
Weight decay	0.05	0.05	0.05	0.05
Base learning rate	1×10^{-4}	1×10^{-4}	5×10^{-5}	5×10^{-5}
Embedding-table LR multiplier	$10\times$	$10\times$	$10\times$	$10\times$
LR schedule	cosine decay	cosine decay	cosine decay	cosine decay
Warmup steps	0	0	0	0
Gradient clip	1.0	1.0	1.0	1.0
Mixed precision	bf16	bf16	bf16	bf16
Adaptation budget				
Pretraining length (epochs)	300	100	400	400
T_{ft} (epochs)	100	25	100	50
T_{ft} as % of pretraining	33.3%	25.0%	25.0%	12.5%
S_{ramp} (epochs)	25	10	30	25
Sequence length (tokens/image)	256	576	256	256
Future-aware embedding				
Embedding dim d	matches model	matches model	matches model	matches model
Initialization	zero	zero	RAR ckpt	RAR ckpt
Null embedding $E_\varnothing$	learned	learned	learned	learned
Sampling (evaluation)				
Decoding	top-k	top-k	top-k	top-k
Temperature	1.0	1.0	1.0	1.0
Classifier-free guidance scale	1.75	1.75	paper schedule[‡]	paper schedule[‡]

[‡]For RAR-L/XL, we follow the per-step CFG schedule used by the original RAR paper [5] for a like-for-like comparison.

Parallelism. Adaptation uses standard data-parallel training; we did not use tensor or pipeline parallelism, since both base-model families fit on a single GPU at the batch sizes we use. Activation checkpointing is enabled for the XL variants. We use bfloat16 mixed precision throughout, with float32 master weights for the optimizer state.

Permutation sampling. At each training step s, each example in the global batch independently draws a permutation σ from the curriculum mixture $P_{\alpha(s)}$ defined in Equation 3: with probability $1 - \alpha(s)$ the raster-scan order is used, and with probability $\alpha(s)$ a fresh uniform random permutation over the n image-token positions is sampled. We do not pre-sample permutations or share them across examples in a batch.

Evaluation protocol. We follow the standard ImageNet 256×256 class-conditional evaluation protocol: $50{,}000$ samples generated under the same number of decoding steps as the corresponding pretrained model (Table 8, "Steps" column in Table 1), with FID, IS, and Precision/Recall computed against the standard ImageNet reference statistics. For inpainting and outpainting, masks are applied identically across methods being compared, and FID, IS, and Precision/Recall are reported over the full image (PSNR and SSIM in Table 4 are computed over the generated region).

D Adaptation Algorithm

Algorithm 1 gives the per-step adaptation loop in pseudocode. The procedure is identical to standard causal AR fine-tuning except for three additions: the per-example permutation σ drawn from the curriculum mixture $P_{\alpha(s)}$ (line 5), the spatial positions $\sigma(t)$ passed into the model's positional encoding (rotary for LlamaGen, additive learned absolute for RAR) so that positional information reflects image geometry rather than sequence step (line 11), and the future-aware embedding $E_{\sigma(t+1)}$ added to each input position before the transformer forward pass (line 13).

Algorithm 1 Adaptation step for unlocking any-order generation in a pretrained AR model.

Require: pretrained causal AR model f_θ with a positional encoding indexed by spatial position (e.g., RoPE or additive learned absolute)
Require: future-aware embedding table $E \in \mathbb{R}^{n \times d}$, initialized to zero (loaded from the checkpoint for RAR)
Require: null embedding $E_\varnothing \in \mathbb{R}^d$
Require: curriculum schedule $\alpha(s) = \min(s/S_{\text{ramp}}, 1)$
Require: number of adaptation steps T_{ft}

for $s = 1$ **to** T_{ft} **do**
  sample minibatch of images, tokenize each into $x = (x_1, \ldots, x_n)$
  for each example in the minibatch **do**
    $u \sim \text{Uniform}[0, 1]$
    if $u < \alpha(s)$ **then**
      $\sigma \leftarrow$ uniform random permutation of $(1, \ldots, n)$
    else
      $\sigma \leftarrow (1, 2, \ldots, n)$ ▷ raster-scan
    end if
    reorder tokens: $\tilde{x} \leftarrow (x_{\sigma(1)}, \ldots, x_{\sigma(n)})$
    spatial position vector: $\mathbf{p} \leftarrow (\sigma(1), \sigma(2), \ldots, \sigma(n))$ ▷ passed to positional encoding
    compute base input representations h_t from $\tilde{x}_{<t}$ as in standard AR
    for $t = 1$ **to** n **do**
      if $t < n$ **then**
        $\tilde{h}_t \leftarrow h_t + E_{\sigma(t+1)}$
      else
        $\tilde{h}_t \leftarrow h_t + E_\varnothing$ ▷ no successor at the final step
      end if
    end for
    predict logits $\ell_t \leftarrow f_\theta(\tilde{h}_{1:t}; \mathbf{p}_{1:t})$ for each t ▷ positional encoding uses spatial $\mathbf{p}$
    per-example loss: $\mathcal{L}_\sigma \leftarrow -\sum_{t=1}^{n} \log p(x_{\sigma(t)} \mid \ell_t)$
  end for
  $\mathcal{L} \leftarrow$ mean of per-example losses over the minibatch
  backpropagate; update θ at base learning rate, E and $E_\varnothing$ at $10\times$ base learning rate
end for

Inference. Algorithm 2 describes generation under an arbitrary order σ. The order can be raster-scan (recovering the pretrained model's behavior), uniform random, or task-induced (observed tokens placed first in σ, with the remainder generated in any chosen order). Because the model is a standard causal single-stream transformer, the keys and values of generated tokens are cached and reused exactly as in standard AR inference; the future-aware embedding only affects the input representation at the current step.

E Broader Impacts

Our recipe extends the capabilities of existing pretrained class-conditional image generators rather than producing a new class of generative model. It inherits both their positive applications, such as image editing and content completion, and their misuse risks, such as misleading content generation. Because the base models are already publicly released and restricted to ImageNet classes, the recipe does not introduce new categories of risk beyond what those models already enable.

F Licenses for Existing Assets

Base models. We adapt publicly released pretrained checkpoints from LlamaGen [1] and RAR [5]. LlamaGen weights are released under the MIT License via the official repository; RAR weights are released under the Apache License 2.0 via the official repository. Both license terms permit research use, redistribution, and the publication of derived models, which covers the adapted checkpoints.

Algorithm 2 Any-order generation with the adapted model.

Require: adapted model f_θ, embedding table E, null embedding $E_\varnothing$
Require: generation order σ (a permutation of $(1, \ldots, n)$)
Require: optional observed tokens $\{x_j\}_{j \in \mathcal{O}}$ (e.g., from an inpainting mask)
place observed positions first in σ if any
spatial position vector: $\mathbf{p} \leftarrow (\sigma(1), \sigma(2), \ldots, \sigma(n))$ ▷ passed to positional encoding at every step
for $t = 1$ **to** n **do**
  if $\sigma(t) \in \mathcal{O}$ **then**
    $x_{\sigma(t)} \leftarrow$ observed value
  else
    compute base input h_t from previously placed tokens $x_{\sigma(<t)}$
    if $t < n$ **then**
      $\tilde{h}_t \leftarrow h_t + E_{\sigma(t+1)}$
    else
      $\tilde{h}_t \leftarrow h_t + E_\varnothing$
    end if
    $\ell_t \leftarrow f_\theta(\tilde{h}_{1:t};\, \mathbf{p}_{1:t})$ ▷ positional encoding uses spatial $\mathbf{p}$; KV cache reused
    $x_{\sigma(t)} \sim \mathrm{softmax}(\ell_t / \tau)$ ▷ or top-k / top-p
  end if
end for
return image tokens $(x_1, \ldots, x_n)$, decode with the base-model VQ tokenizer

Tokenizers. The VQ tokenizers we use are released alongside their corresponding base models and inherit the same licenses. We use them unmodified.

Dataset. Adaptation and evaluation use ImageNet (ILSVRC-2012) [39], accessed under the standard research-use terms published by the dataset maintainers. We do not redistribute any ImageNet imagery; only metric-level results derived from generated samples are reported.

Software. We use `vLLM` [25] (Apache 2.0) for inference benchmarks. All other software dependencies (PyTorch, HuggingFace utilities) are used under their standard open-source licenses.

Released artifacts. Released adapted checkpoints inherit the more restrictive of the base-model licenses they derive from and will be accompanied by a model card documenting training data provenance, intended use, and known limitations.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction (Section 1) state the two contributions of the paper—the future-aware positional embedding and the ordering curriculum—and the central empirical claims (adaptation cost between $1.13\times$ and $1.33\times$ pretraining depending on base model, vs. $\approx 3\times$ from scratch; a measured class-conditional FID trade-off of 1.17–1.93 across four base models; compatibility with the AR inference ecosystem, including a documented path to vLLM serving). These claims are validated quantitatively in Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A dedicated limitations paragraph appears at the end of Section 5, covering scope (ImageNet class-conditional only; non-token-based AR variants not evaluated), the narrow scope of the quantitative editing evaluation (arbitrary-region completion only, not instruction-following or identity preservation), and the caveat on the matched-budget pretraining-value control in Section 4.2 (it does not isolate initialization from the curriculum schedule).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include formal theorems or proofs. The arguments in Section 3 and Appendix A are informal diagnoses (identifiability and gradient-magnitude reasoning) used to motivate empirical design choices, and they are validated experimentally in Section 4.4 rather than proved.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3 fully specifies the recipe (future-aware embedding construction, curriculum schedule, training objective, inference procedure). Section 4.1 reports base models, datasets, fine-tuning budgets, and curriculum hyperparameters. Per-model hyperparameters are tabulated in Appendix B. All base models (LlamaGen-L/XL, RAR-L/XL) are publicly available pretrained checkpoints, and the dataset is the standard ImageNet 256×256 benchmark.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is released at `https://github.com/Ordered-AR/unlockedadaptation`, together with adapted LlamaGen-L and LlamaGen-XL checkpoints. The dataset (ImageNet) is publicly available, the base-model checkpoints we adapt (LlamaGen-L/XL, RAR-L/XL) are publicly released by their respective authors, and the recipe is fully described in Section 3 and Appendix B so that an independent re-implementation is feasible.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.1 specifies base models, dataset, adaptation budget, curriculum schedule, and the future-aware embedding learning-rate multiplier. Per-base-model optimizer settings, learning rates, adaptation epochs, curriculum ramps, and the batch-size policy

are given in Appendix B. The evaluation protocol (FID, IS, Precision/Recall on $50{,}000$ samples, ImageNet protocol) is stated in the same section.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We follow the standard practice in the class-conditional ImageNet generative-modeling literature and report single-run FID/IS/Precision/Recall over $50{,}000$ samples per configuration. Per-checkpoint evaluation variance for these metrics is well below the gaps between methods we report; running multiple full adaptation runs per base model would substantially exceed our compute budget.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Adaptation experiments were run on internal clusters using NVIDIA H200 and NVIDIA RTX PRO 6000 Blackwell Server Edition GPUs. Node configurations and per-model adaptation epoch budgets are reported in Appendix B; LlamaGen-XL adaptation measured $\approx$ 64 H200-GPU-hours (one $8\times$H200 node for $\approx$ 8 hours), reported in Appendix C. The total-budget overhead relative to pretraining ranges from $1.13\times$ (RAR-XL) to $1.33\times$ (LlamaGen-L) depending on base model (Section 4.5); it is not a single value applied uniformly across the four base models.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research adheres to the NeurIPS Code of Ethics. The work uses publicly released pretrained checkpoints and the standard ImageNet benchmark, involves no human subjects or new data collection, and we discuss potential misuse considerations in Appendix E.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Our recipe extends the capabilities of existing pretrained class-conditional image generators rather than producing a new class of generative model; it inherits both the positive applications (image editing, content completion) and the misuse risks (e.g., misleading content generation) of the underlying base models. Because the base models are already publicly released, our recipe does not introduce new categories of risk beyond what those models already enable. We discuss this in Appendix E.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: We release adapted LlamaGen-L and LlamaGen-XL checkpoints. Because our adapted models are class-conditional ImageNet generators that inherit the capabilities of publicly released base models (LlamaGen, RAR), the misuse surface is bounded by what the base models already enable. We will accompany the released checkpoints with documentation describing intended use, training data provenance, and known limitations.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All base models and datasets we use are publicly released and are cited in the main paper. Specifically, we use LlamaGen [1] and RAR [5] pretrained checkpoints under their respective released licenses, and the ImageNet dataset under its standard research-use terms. The associated VQ tokenizers used by each base model are released alongside the base models and used unmodified. License details for each asset are given in Appendix F.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

Justification: We release the code (`https://github.com/Ordered-AR/unlockedadaptation`) and adapted LlamaGen-L and LlamaGen-XL checkpoints. The accompanying documentation will include a model card describing the recipe, training data, intended use, evaluation results, and known limitations.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects. All evaluation is automated using standard ImageNet generative-modeling metrics (FID, IS, Precision/Recall).

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human subjects, so IRB approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Large language models are not part of the core methodology of this paper. Any LLM use in the project was limited to writing assistance (editing prose, polishing phrasing), which the NeurIPS LLM policy explicitly exempts from declaration.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.