SCANAR: PARALLEL ROW-SCAN DECODING FOR EFFICIENT IMAGE GENERATION IN UNIFIED AUTOREGRESSIVE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Unified multimodal autoregressive (Unified AR) models represent text and images in a shared token sequence and generate all modalities with a common next-token prediction objective. However, this raster-order decoding requires HW sequential decisions for an $H \times W$ image, overlooking the inherent two-dimensional structure of visual tokens and creating a major inference bottleneck. To address this problem, we propose ScanAR, a row-scan parallel autoregressive method for efficient image generation within Unified AR models. Instead of modifying the generation head or introducing additional parameters, ScanAR accelerates decoding by redefining the predecessors of visual tokens, enabling seamless integration with existing Unified AR models. Specifically, the first visual row follows the standard left-to-right dependency, while each subsequent row is generated in parallel by conditioning each token on its vertically aligned predecessor in the previous rows. For nonvisual tokens, the standard autoregressive rule is unchanged. In this way, ScanAR reduces the sequential decoding from HW to $H + W$, supporting both training from scratch and lightweight post-training without modifying the backbone, vocabulary, or output head. Extensive experiments on Emu3.5 demonstrate that ScanAR improves the GenEval score by 1.88 points and achieves a $27.7\times$ speedup for 512×512 image generation through post-training. On U0, ScanAR accelerates image generation while retaining its interleaved and video generation capabilities.

1 INTRODUCTION

The combination of discrete visual tokenizers and autoregressive Transformers casts image generation as a sequence modeling problem analogous to language modeling Esser et al. (2021); Wang et al. (2024). Unified AR models further place textual and visual tokens in a shared vocabulary under a common next-token prediction interface, enabling a single model to generate arbitrary mixed-modal sequences Chameleon Team (2024); Wang et al. (2024). Visual content, however, typically corresponds to a large two-dimensional token grid. Flattening this grid and decoding it token by token concentrates most of the sequential execution time of a unified generator in its visual spans. For an image of size $H \times W$, standard raster AR requires HW visual-token decisions, resulting in substantial latency and throughput constraints at high resolutions.

Recent approaches to accelerating visual AR generation include scale-wise generation restructuring, spatial generation-path restructuring, and speculative decoding Tian et al. (2024); Wang et al. (2025b); He et al. (2025); Zhou et al. (2026); Teng et al. (2025). Most existing methods center on a prespecified single-image canvas and formulate acceleration as a relatively self-contained visual decoding process. Their applicability to Unified AR models is limited. A general image acceleration method for Unified AR should instead introduce visual parallelism within the same conditional model that predicts text and image structure, while remaining applicable to visual segments throughout the unified sequence.

To address this limitation, we propose ScanAR, an image-generation acceleration method that preserves Unified AR capabilities. ScanAR derives row-scan parallelism directly from the two-dimensional structure of image tokens, while all modalities remain modeled by a shared next-token

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.

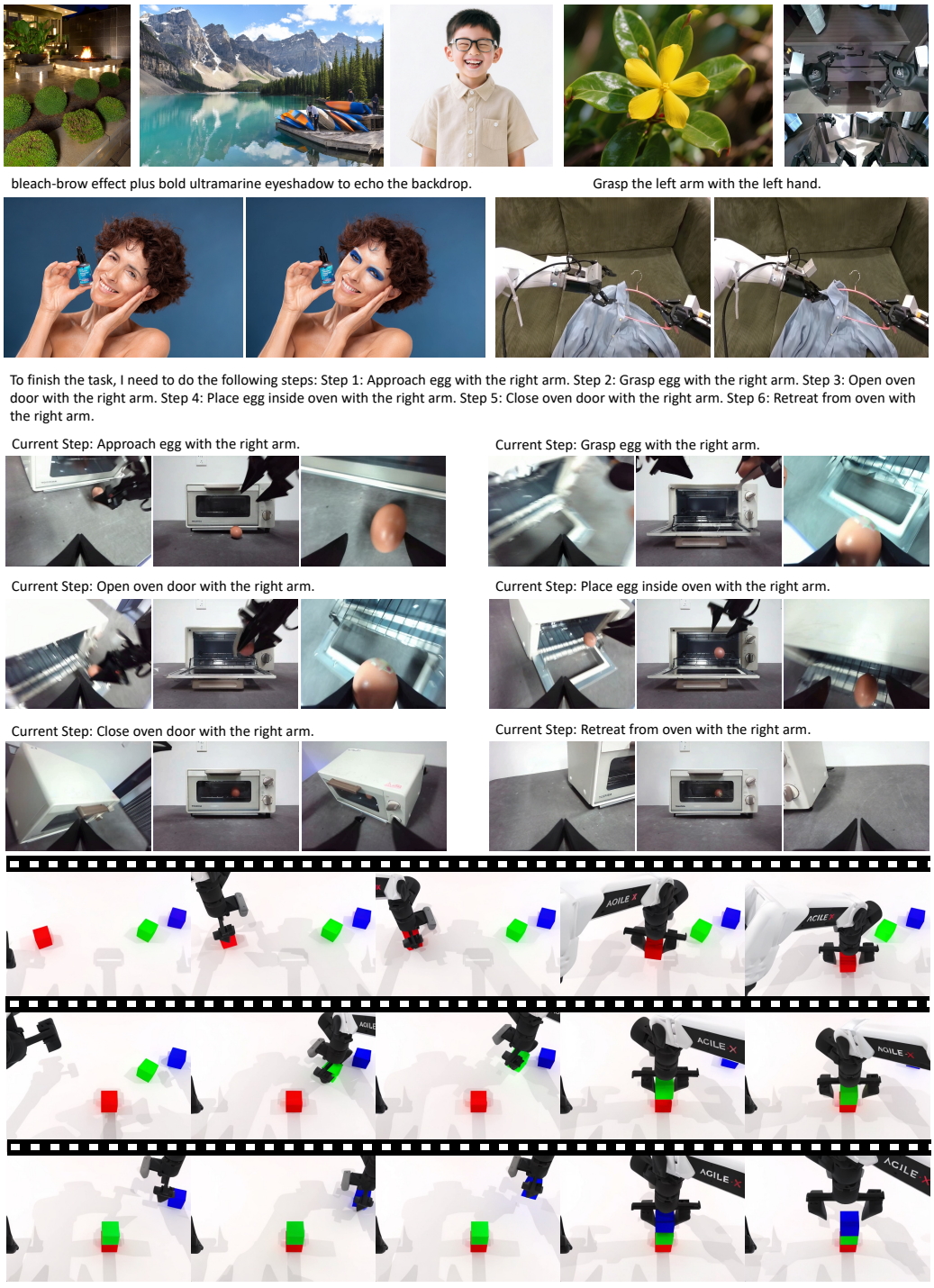


Figure 1: Qualitative results of ScanAR across diverse generation tasks, including text-to-image generation, image editing, interleaved image-text generation, and video generation.

prediction head. Text tokens, image-structure tokens, and tokens in the first visual row use their standard causal predecessors, whereas each subsequent visual token uses the hidden state at the same column in the preceding row as its predecessor. This predecessor relation groups all positions in each later row into a single parallel prediction step, while rows advance autoregressively in row-major order. During training, a mixed attention mask and a joint objective enable standard causal and row-scan dependencies to be learned within the same prediction space. Extensive experiments demonstrate that models adapted with ScanAR post-training retain or even surpass their original capabilities. On Emu3.5 Cui et al. (2025a), ScanAR post-training improves the GenEval score by 1.88% and achieves a $27.7\times$ speedup for 512×512 image generation. On U0 Li et al. (2026), ScanAR post-training preserves its interleaved and video-generation capabilities, as shown in Figure 1. Our main contributions are as follows:

- We introduce a two-dimensional row-scan decoding scheme that predicts each subsequent visual row in parallel, reducing the sequential decoding depth of an $H \times W$ visual grid from $H \times W$ to $H + W$ and achieving a $27.7\times$ generation speedup at 512×512 resolution.
- We preserve raster spatial indexing and row-major progression without introducing additional parameters, enabling lightweight post-training adaptation of raster AR models as well as training from scratch.
- We retain the Unified AR model structure, accelerating visual segments while preserving the Unified AR capability to generate text, images, and interleaved text–image sequences.

2 Related Work

2.1 Unified Autoregressive Models

Unified AR models, exemplified by Chameleon and Emu3, represent text and images as a unified sequence of discrete tokens and model their conditional distribution with a common autoregressive objective Cui et al. (2025a); Chameleon Team (2024); Wang et al. (2024). For image generation, a continuous image is typically encoded by a visual tokenizer into a two-dimensional token grid and then inserted into the unified sequence in a prescribed order. In mixed-modal generation, textual tokens and visual tokens share the same context, Transformer backbone, and output interface. This paradigm offers a clear probabilistic formulation, inherits next-token prediction and scaling principles from language modeling, and structurally preserves the ability to generate text, images, and interleaved multimodal content within a single model.

The principal bottleneck of Unified AR models lies in their image segments. A standard implementation flattens the two-dimensional visual grid into a raster sequence and generates it token by token in a fixed order. Although this preserves a strict causal factorization, it has two notable limitations. First, the two-dimensional spatial neighborhood is compressed into a one-dimensional ordering, forcing the model to express local image dependencies through a fixed scan path. Second, the inference depth of an image segment scales linearly with the number of visual tokens and can far exceed that of ordinary text generation at high resolutions. Unified AR models thus possess a native ability to generate both modalities, but the serial decoding of image tokens substantially limits the efficiency of their visual generation capability.

2.2 Acceleration of Visual Autoregressive Generation

Existing approaches to visual AR acceleration can be broadly grouped into scale-wise generation restructuring, spatial generation-path restructuring, and speculative decoding. The first two exploit the structured organization of visual representations: the former changes the prediction granularity, whereas the latter changes the generation topology over spatial positions. The third reduces the effective number of decoding rounds through drafting and verification.

Generation-granularity restructuring reformulates image generation from token-level prediction into scale-level or block-level prediction. For example, VAR casts visual generation as coarse-to-fine next-scale prediction Tian et al. (2024). This formulation substantially reduces the number of sequential stages. However, it requires a redesigned prediction objective and a scale-structured tokenizer. The learned conditional distribution is no longer the standard next-token factorization over

Table 1: Comparison of representative and recent approaches to visual AR inference acceleration.

Method	Decoding depth	No extra parameters	Raster-order progression	Support unified output
Baseline				
Unified raster AR Wang et al. (2024)	HW	✓	✓	✓
Generation-granularity restructuring				
VAR Tian et al. (2024)	10	×	×	×
DenseAR Park et al. (2026)	64	✓	×	×
Spatial-path restructuring				
PAR Wang et al. (2025b)	$HW/p + p$	×	×	×
RandAR Pang et al. (2024)	88	×	×	×
NAR He et al. (2025)	$H + W - 1$	×	×	×
FlashAR Zhou et al. (2026)	$H + W - 1$	×	×	×
LPD Zhang et al. (2026)	$20/48$	×	×	×
RadAR Wang et al. (2025a)	$10/13$	×	×	×
Speculative/Jacobi decoding				
SJD Teng et al. (2025)	Adaptive	✓	✓	×
PJD Liao et al. (2026)	Adaptive	✓	×	×
CSD Wang et al. (2026)	Adaptive	✓	✓	×
SSD Xiang et al. (2026)	n (square)	×	×	×
Ours				
ScanAR (ours)	$H + W$	✓	✓	✓

✓/× denotes yes/no. Numeric depths are reported configurations (RandAR uses 256 visual tokens); symbolic depths omit constant offsets. Here, p is the PAR parallel width and n the side length of a square grid.

the original visual-token sequence. Consequently, such methods are not directly compatible with the Unified AR model.

Spatial generation-path restructuring exploits the two-dimensional structure of visual tokens by reorganizing the raster decoding topology. Representative approaches include token grouping in PAR, frontier expansion in NAR, random-order decoding in RandAR, and directional prediction in FlashAR Wang et al. (2025b); He et al. (2025); Pang et al. (2024); Zhou et al. (2026). However, these methods may disrupt local generation continuity, introduce additional prediction parameters, or require a predefined visual canvas and decoding span, limiting their direct application as acceleration methods for Unified AR models.

Speculative decoding accelerates generation by drafting multiple candidate tokens and verifying them with the target model Leviathan et al. (2023). SJD extends this approach to visual generation using speculative Jacobi decoding Teng et al. (2025). However, its practical efficiency depends on draft quality, acceptance rate, and verification cost, while visual-token correlations and modality boundaries introduce additional computational and system complexity in Unified AR models.

ScanAR uses a deterministic row-frontier dependency that preserves row-major progression while predicting tokens conditioned on the preceding row in parallel. It shares the same multimodal output head for standard causal and row-scan prediction, preserving a unified next-token space across modalities. Applied independently to each visual segment, it supports dynamic image sizes and general Unified AR outputs. Table 1 summarizes the distinctions among these acceleration paradigms.

3 Method

3.1 Unified Prediction Space

Given a multimodal prefix x, a unified autoregressive (AR) model generates a sequence z containing text, control tokens, and a variable number of image blocks. Image boundaries and spatial dimensions are generated as structural tokens. ScanAR accelerates each visual segment through row-scan prediction while retaining the original backbone, vocabulary, and shared next-token prediction head.

Text and structural tokens retain their standard causal predecessors. Visual tokens use the predecessor mapping defined below, independently for each image block. This formulation supports arbitrary image positions and dimensions within a unified output stream.

3.2 Row-Scan Prediction

For an $H \times W$ visual grid, let $y_{i,j}$ denote the token at row i and column j, using zero-based indices. ScanAR assigns each token to the generation step

$$s(i,j) = \begin{cases} j, & i = 0, \\ W + i - 1, & i > 0. \end{cases} \tag{1}$$

The first row is generated from left to right, and each later row forms one parallel group. Equation equation 1 counts $H + W - 1$ sampling steps. For $H = 1$, an additional state-only first-row commit gives an effective visual decoding depth of $H + W$, compared with HW for raster AR; a single-row grid uses W steps.

Let $h^{\text{causal}}_{\text{pref}}$ denote the causal hidden state of the image prefix. This state is used to predict the first visual token. For the first row, let $h^{\text{causal}}_{0,j}$ denote the hidden state of visual token $y_{0,j}$ after it has been committed under causal attention. Let $\bar{h}_{i,j}$ denote the hidden state at position (i,j) after the complete row i has been submitted with full visibility within that row.

The prediction state for visual token $y_{i,j}$ is defined as

$$g_{i,j} = \begin{cases} h^{\text{causal}}_{\text{pref}}, & i = 0,\ j = 0, \\ h^{\text{causal}}_{0,j-1}, & i = 0,\ j > 0, \\ \bar{h}_{i-1,j}, & i > 0. \end{cases} \tag{2}$$

Thus, the tokens in the first row are predicted using causal hidden states. Starting from the second row, the token at column j uses the fully visible hidden state of the token at the same column in the preceding row.

Since the first row is generated causally, the hidden state of each first-row token only incorporates the preceding tokens. These causal hidden states cannot directly serve as the predecessors for the second row, whose tokens should condition on the complete first row.

After the first row has been generated, ScanAR performs one non-sampling first-row commit operation. The generated first-row tokens are reprocessed together with the image prefix, while all tokens within the first row are made mutually visible. This operation produces the fully visible states $\bar{h}_{0,j}$, which are then used to predict the second row.

No separate commit operation is required for later rows. From the second row onward, all tokens in the current row are submitted to the backbone as a group. During this forward pass, every query in the current row can attend to all tokens in that row, in addition to the prefix and all previously completed rows. Therefore, the resulting hidden states are already the fully visible states $\bar{h}_{i,j}$ and can be used directly as the predecessors for row $i + 1$.

Accordingly, the decoding schedule consists of W causal generation steps for the first row, one non-sampling commit step, and one parallel generation step for each subsequent row. The generated tokens within a parallel row are conditionally independent given the image prefix and all completed preceding rows, while the row-level autoregressive dependency is preserved.

3.3 Attention Mask and Training

The mask specifies hidden-state context, and Equation equation 2 selects the state used to predict each target. Let R_i denote the positions of visual row i and b_i its first position. For queries and keys in the original sequence, the visibility set is

$$\mathcal{V}(q) = \begin{cases} \{k \mid k \le q\}, & q \text{ is nonvisual or } q \in R_0, \\ \{k \mid k < b_i\} \cup R_i, & q \in R_i,\ i > 0. \end{cases} \tag{3}$$

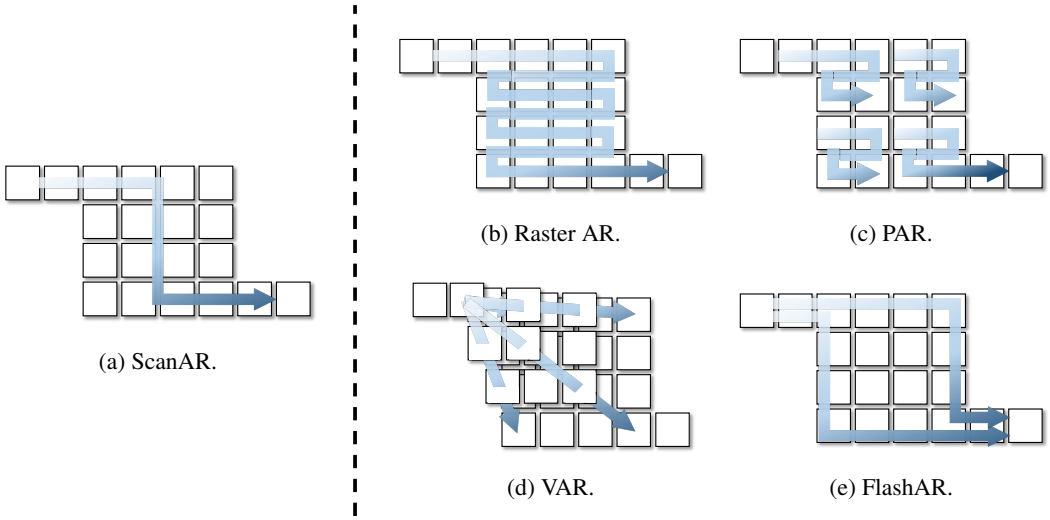


Figure 2: Comparison of five different AR decoding methods. ScanAR uses a concise row-scan design that remains compatible with multimodal AR generation.

Nonvisual positions and the first row use causal attention. Later rows use full within-row attention to form committed states for subsequent predictions.

Training uses ground-truth tokens throughout. An auxiliary copy of the first row, with identical token and position IDs, computes $\bar{h}_0$ in the same forward pass as the original sequence. Its queries attend only to the preceding context and their own complete copy; original sequence positions attend only to original positions. The copy has no independent prediction targets and receives gradients through the second-row loss. Figure 3 illustrates the mask and predecessor mapping.

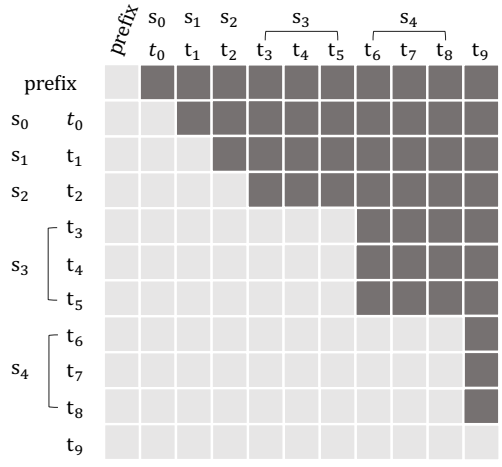


Figure 3: Training visibility and predecessor mapping of ScanAR over a unified sequence.

For each image, visual cross-entropy is evaluated on the logits obtained from the selected predecessor states:

$$\mathcal{L}_{\text{vis}} = -\frac{1}{HW} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \log \text{softmax}(\ell_{i,j})_{y_{i,j}}. \tag{4}$$

Text and structural targets use standard preceding-position cross-entropy. The joint objective optionally includes visual distribution alignment with a frozen raster-AR teacher Hinton et al. (2015):

$$\mathcal{L} = \mathcal{L}_{\text{vis}} + \lambda_{\text{txt}} \mathcal{L}_{\text{txt/ctrl}} + \lambda_{\text{d}}(t) T^2 \, \text{KL}\left(p_{\text{AR}}^{(T)} \,\|\, p_{\text{Scan}}^{(T)} \right), \tag{5}$$

where the KL term is averaged over aligned visual targets at temperature T. Both models receive the ground-truth sequence: the teacher uses raster-causal predecessors, and the student uses $g_{i,j}$. During post-training, $\lambda_{\mathrm{d}}(t)$ is annealed to zero after initial alignment; it is zero when training from scratch. ScanAR supports both regimes without adding trainable parameters.

3.4 Unified Inference and KV Cache

Inference evaluates the same predecessor mapping using sampled tokens. First-row tokens are sampled and appended to the cache individually. The first-row commit then recomputes all W positions through every decoder layer with full within-row attention, reusing the preceding context's KV. The resulting $\bar{h}_0$ is stored separately for second-row prediction; temporary commit KV is discarded, and the main cache retains the original causal first-row KV.

For each later row, ScanAR samples all tokens from the preceding row's committed states, then processes the sampled row through every decoder layer with full within-row attention. This forward pass produces its committed states and appends its layer-wise KV at the original sequence positions. Earlier cache entries remain fixed.

After the image, text and structural tokens resume standard causal decoding over the accumulated cache, which contains the causal first-row KV and the subsequent rows' KV computed with within-row attention. The separate first-row commit states supply prediction sources without adding output tokens or replacing the sequence cache.

4 Experiments

4.1 Experimental Setup

We evaluate ScanAR in two settings. The primary setting uses the 34B Xiaomi-Robotics-U0 unified multimodal autoregressive model Li et al. (2026). The secondary setting follows the image-only protocols used by prior work and covers both class-conditional and text-to-image generation. Results from the two settings are reported separately because they use different base models, datasets, and evaluation protocols.

For the unified multimodal setting, ScanAR is initialized from a multimodal Xiaomi-Robotics-U0 checkpoint and retains the original visual tokenizer, unified vocabulary, Transformer backbone, and output head. We perform full-parameter post-training without adding trainable parameters. The training corpus contains image-generation and interleaved multimodal data, including image editing, subject preservation, multi-reference conditioning, spatial control, embodied-scene generation and transfer, video generation, and interleaved image-text generation. Training uses visual-token cross-entropy together with distribution-level supervision from a frozen raster-AR teacher during the initial adaptation stage. Detailed settings are given in Appendix A.1.

For direct comparison with prior work, the image-only experiments follow the protocol of FlashAR Zhou et al. (2026). They evaluate LlamaGen Sun et al. (2024) on class-conditional ImageNet generation Russakovsky et al. (2015) and Emu3.5-Image-34B Cui et al. (2025a) on text-to-image synthesis. Their protocols are described separately in Appendix A.2.

4.2 Generation Quality

Unified multimodal generation.

We compare the original Xiaomi-Robotics-U0 model with its ScanAR-adapted counterpart on embodied-scene transfer. This task tests whether accelerated visual decoding preserves the model's conditional scene-generation capability under depth and multi-view constraints.

We use GPT-5.4 as an automated evaluator. Given the scene instruction, depth maps, and generated multi-view images, it scores six dimensions on a 10-point scale: depth faithfulness, background alignment, foreground alignment, lighting, image quality, and multi-view consistency. The results are shown in Table 2.

ScanAR obtains an overall score of 7.49, close to the original model's 7.52. It improves depth faithfulness from 7.4 to 7.5 and multi-view consistency from 7.5 to 7.6, while preserving the lighting score at 7.5. The changes in the remaining dimensions are small, indicating that row-scan decoding preserves the quality of embodied-scene transfer. The details of the scoring can be found in Appendix B.1. Additional results on interleaved generation and image generation are provided in Appendix B.2.

Table 2: GPT-5.4 evaluation results on embodied-scene transfer using U0. Scores are reported on a 10-point scale, with higher values indicating better performance.

Method	Depth Faithfulness	Background	Foreground	Lighting	Image Quality	Multi-View Consistency	Overall
U0	7.4	7.6	7.3	7.5	8.0	7.5	7.52
ScanAR	7.5	7.4	7.2	7.5	7.8	7.6	7.49

Class-conditional generation. To demonstrate ScanAR's model-fitting capability, we train LlamaGen-based ScanAR models at four different scales from scratch. Table 3 reports class-conditional ImageNet generation at 256×256. ScanAR achieves the highest Inception Score (IS) and throughput at every model size. Its IS increases from 244.3 for the B model to 306.9 for the XXL model, while its throughput remains higher than that of the other methods. Additional results are provided in Appendix B.3.

Table 3: Inception Score and throughput on the ImageNet 256×256 benchmark. Each entry reports IS / throughput (img/s).

Method	Type	Epoch	B ($\sim$ 120M)	L ($\sim$ 360M)	XL ($\sim$ 700M)	XXL ($\sim$ 1.4B)
LlamaGen	From scratch	300	193.6 / 117.9	248.3 / 47.1	227.1 / 23.7	253.6 / 14.1
PAR	From scratch	300	204.4 / 174.1	189.4 / 93.8	234.4 / 53.9	288.3 / 33.9
NAR	From scratch	300	212.3 / 419.7	263.9 / 195.4	277.5 / 98.1	293.5 / 56.9
BlockDiffusion	Post-training	75	176.2 / 186.3	243.5 / 103.2	258.6 / 41.7	264.9 / 26.8
FlashAR	Post-training	25	208.3 / 447.2	289.0 / 224.7	293.7 / 109.3	289.4 / 63.4
ScanAR	From scratch	300	**244.3 / 669.4**	**295.2 / 235.2**	**301.8 / 113.4**	**306.9 / 65.3**

Text-to-image generation.

We also evaluate ScanAR on Emu3.5-Image-34B. Table 4 jointly reports post-training cost, inference latency, and generation quality. ScanAR uses 20K post-training steps on 80M training tokens, compared with 50K steps for the other post-training baselines. It reduces the serial visual decoding depth from 1024 to 64 and achieves a $27.7\times$ wall-clock speedup for 512×512 generation. Its overall GenEval score is 82.36, exceeding the original autoregressive model's score of 80.48. Training-cost and decoding-step details are given in Appendix B.4.

Table 4: Training cost, inference efficiency, and overall GenEval at 512×512 resolution.

Method	Type	Steps	Data	Latency (s)↓	GenEval↑
Emu3.5-Image	From scratch	940K	150B	130.10	80.48
BlockDiffusion	Post-training	50K	80M	6.17	73.83
FlashAR	Post-training	50K	80M	5.68	80.29
ScanAR	**Post-training**	**20K**	**80M**	**4.69**	**82.36**

The detailed GenEval results are reported in Table 5. ScanAR achieves the best overall score of 82.36, exceeding the original Emu3.5-Image model by 1.88 points, FlashAR by 2.07 points, and BlockDiffusion by 8.53 points. The largest gain is observed in Counting, where ScanAR reaches 69.06 compared with 53.75 for both the original model and FlashAR. It also obtains 80.00 on Position and 99.06 on Single Object, while the category-level scores vary across methods.

Table 5: GenEval scores on Emu3.5-Image-34B at 512×512 resolution.

Method	Overall	Single Obj.	Two Obj.	Counting	Colors	Position	Color Attr.
Emu3.5-Image	80.48	**100.00**	**94.95**	53.75	90.96	73.00	**70.25**
BlockDiffusion	73.83	96.88	88.89	47.50	85.64	68.00	58.44
FlashAR	80.29	98.75	91.92	53.75	**92.55**	80.00	64.00
ScanAR	**82.36**	99.06	87.88	**69.06**	90.16	**80.00**	68.00

Across the unified multimodal and image-only settings, these results show that ScanAR preserves generation quality while reducing the sequential cost of visual decoding. The following experiments isolate this efficiency advantage in training and serving.

4.3 Training and Inference Efficiency

ScanAR introduces no additional trainable parameters and retains the AR model components. Its inference schedule uses only single-token decoding and full-row decoding, reducing visual decoding depth while preserving a regular execution pattern.

With block-sparse attention, ScanAR maintains training efficiency close to the causal AR baseline. Detailed training measurements are provided in Appendix C. For inference, we evaluate vLLM-based implementations on a single H20E GPU at 512×512 resolution. After warm-up, 64 requests are submitted concurrently. The reported time per image is derived from aggregate throughput and therefore represents amortized generation time rather than single-request latency. The results are shown in Table 6.

Table 6: Inference efficiency of vLLM-based implementations on a single H20E GPU at 512×512 resolution. After warm-up, 64 requests are submitted concurrently. Time per image is derived from aggregate throughput across all requests.

Method	Time per image (s) ↓	Throughput (images/s) ↑	Peak memory (GB) ↓
FlashAR	1.37	0.73	138.8
ScanAR	1.15	0.87	110.7

ScanAR reduces the amortized time per image from 1.37 to 1.15 seconds and increases aggregate throughput from 0.73 to 0.87 images per second, corresponding to a 19.1% improvement over FlashAR. Peak memory decreases from 138.8 to 110.7 GB, a reduction of 28.1 GB. The regular row-wise schedule enables predictable execution and efficient reuse of serving kernels.

5 Conclusions

We presented ScanAR, a row-scan parallel autoregressive method that accelerates image generation in Unified AR models without modifying the backbone, vocabulary, or output head, or introducing additional parameters. ScanAR preserves the shared token sequence and common next-token prediction objective, leaving the standard autoregressive dependencies unchanged for nonvisual tokens and the first visual row. For each subsequent visual row, it redefines the predecessor of every visual token as its vertically aligned counterpart in the preceding row, enabling parallel prediction within the row while retaining a unified AR formulation. This predecessor redesign reduces the sequential decoding depth for an $H \times W$ image from HW to $H + W$ and applies independently to each image segment in an interleaved multimodal sequence. ScanAR supports both training from scratch and lightweight post-training on existing Unified AR models. On Emu3.5, it improves the GenEval score by 1.88 points and achieves a $27.7\times$ speedup for 512×512 image generation through post-training. On U0, ScanAR accelerates image generation while preserving interleaved and video generation capabilities. These results demonstrate that ScanAR provides a simple and practical approach to efficient image generation within Unified AR models.

AI USE STATEMENT

In this work, we used generative AI tools solely for language editing, including grammar correction, proofreading, and improvements to the clarity and readability of the manuscript. We have not used generative AI tools for code development or modification, experiment design or execution, result verification, performance analysis, interpretation of findings, or the generation of figures, tables, and other research artifacts, and other disclosure categories are not applicable to this work. Additionally, we did not use generative AI tools for any other research-related tasks. We have reviewed all AI-assisted work. In particular, we carefully checked the polished text against the original manuscript, experimental results, and supporting evidence to ensure that it is accurate, faithful to the intended meaning, and free of unsupported claims. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024. URL https://arxiv.org/abs/2405.09818.

Yushi Chen et al. Opengpt-4o-image. *arXiv preprint*, 2025a.

Yushi Chen et al. Sharegpt-4o-image. *arXiv preprint*, 2025b.

Yufeng Cui, Honghao Chen, Haoge Deng, Xu Huang, Xinghang Li, Jirong Liu, Yang Liu, Zhuoyan Luo, Jinsheng Wang, Wenxuan Wang, et al. Emu3. 5: Native multimodal models are world learners. *arXiv preprint arXiv:2510.26583*, 2025a.

Yufeng Cui et al. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025b. URL https://arxiv.org/abs/2503.09573.

Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12873–12883, 2021. URL https://arxiv.org/abs/2012.09841.

Dhruba Ghosh, Hannaneh Hajishirzi, et al. Geneval: An object-focused framework for evaluating text-to-image alignment. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2310.11513.

Yefei He, Yuanyu He, Shaoxuan He, Feng Chen, Hong Zhou, Kaipeng Zhang, and Bohan Zhuang. Neighboring autoregressive modeling for efficient visual generation. *arXiv preprint arXiv:2503.10696*, 2025. URL https://arxiv.org/abs/2503.10696.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, Thomas ayed, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems*, 30, 2017. URL https://arxiv.org/abs/1706.08500.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. URL https://arxiv.org/abs/1503.02531.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. URL https://arxiv.org/abs/2211.17192.

Xinghang Li, Jun Guo, Qiwei Li, Long Qian, Hang Lai, Yueze Wang, Hongyu Yan, Jiahang Cao, Xi Chen, Jingen Qu, Jiaxi Song, Nan Sun, Hanye Zhao, Futeng Liu, Wanli Peng, Heyun Wang, Yunhong Wang, Caoyu Xia, Jack Zhao, and Diyun Xiang. Xiaomi-Robotics-U0: Unified embodied synthesis with world foundation model. *arXiv preprint arXiv:2607.11643*, 2026. URL https://arxiv.org/abs/2607.11643.

Boya Liao, Ying Li, Siyong Jian, and Huan Wang. Parallel jacobi decoding for fast autoregressive image generation. *arXiv preprint arXiv:2606.05703*, 2026. URL https://arxiv.org/abs/2606.05703.

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. *arXiv preprint arXiv:2412.01827*, 2024. URL https://arxiv.org/abs/2412.01827.

Chicago Y. Park, Jialin Mao, Xiaojian Xu, Taha Kass-Hout, Ulugbek S. Kamilov, and Cao Xiao. Next-dense-stride prediction for multimodal autoregressive visual modeling. *arXiv preprint arXiv:2607.09892*, 2026. URL https://arxiv.org/abs/2607.09892.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. In *International Journal of Computer Vision*, volume 115, pp. 211–252, 2015. URL https://arxiv.org/abs/1409.0575.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, volume 29, 2016. URL https://arxiv.org/abs/1606.03498.

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. LlamaGen: Text-to-image generation with next-token prediction. *arXiv preprint arXiv:2406.06525*, 2024. URL https://arxiv.org/abs/2406.06525.

Yao Teng, Han Shi, Xian Liu, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. Accelerating auto-regressive text-to-image generation with training-free speculative jacobi decoding. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.01699.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *arXiv preprint arXiv:2404.02905*, 2024. URL https://arxiv.org/abs/2404.02905.

Mingcheng Wang, Junbo Qiao, Yunchen Li, Lingfu Jiang, Wei Li, Jie Hu, Jiao Xie, Zhou Yu, Xinghao Chen, Guixu Zhang, and Shaohui Lin. CSD: Content-aware speculative decoding for efficient image generation. *arXiv preprint arXiv:2606.27829*, 2026. URL https://arxiv.org/abs/2606.27829.

Siyang Wang, Hanting Li, Wei Li, Jie Hu, Xinghao Chen, and Feng Zhao. From sequential to spatial: Reordering autoregression for efficient visual generation. *arXiv preprint arXiv:2512.24639*, 2025a. URL https://arxiv.org/abs/2512.24639.

Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, Yingli Zhao, Yulong Ao, Xuebin Min, Tao Li, Boya Wu, Bo Zhao, Bowen Zhang, Liangdong Wang, Guang Liu, Zheqi He, Xi Yang, Jingjing Liu, Yonghua Lin, Tiejun Huang, and Zhongyuan Wang. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024. URL https://arxiv.org/abs/2409.18869.

Yuqing Wang, Shuhuai Ren, Zhijie Lin, Yujin Han, Haoyuan Guo, Zhenheng Yang, Difan Zou, Jiashi Feng, and Xihui Liu. Parallelized autoregressive visual generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025b. URL https://arxiv.org/abs/2412.15119.

Shilong Xiang, Zirui Zhang, Lijun Yu, and Chengzhi Mao. SSD: Spatially speculative decoding accelerates autoregressive image generation. *arXiv preprint arXiv:2606.20543*, 2026. URL https://arxiv.org/abs/2606.20543.

Zhuoyang Zhang, Luke J. Huang, Chengyue Wu, Shang Yang, Kelly Peng, Yao Lu, and Song Han. Locality-aware parallel decoding for efficient autoregressive image generation. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2507.01957.

Junkang Zhou, Yefei He, Feng Chen, Weijie Wang, and Bohan Zhuang. FlashAR: Efficient post-training acceleration for autoregressive image generation. *arXiv preprint arXiv:2605.09430*, 2026. URL https://arxiv.org/abs/2605.09430.

A Detailed Experimental Settings

A.1 Unified Model Setting

The Xiaomi-Robotics-U0 model comprises 64 decoder-only Transformer layers with a hidden dimension of 5,120, an FFN dimension of 25,600, 64 attention heads, and 8 grouped-query attention groups.

We optimize ScanAR for 10K steps with a global batch size of 512 using Adam with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, a weight decay of 0.1, and gradient clipping at 5.0. The learning rate is linearly warmed up to 1×10^{-5} during the first 1% of training and then decayed to 1×10^{-6} using a cosine schedule.

During the first 1K steps, a frozen raster-scan autoregressive teacher provides distribution-level supervision at all target visual positions with a distillation temperature of 2.0. The distillation weight is linearly annealed from 0.3 to 0, while the visual-token cross-entropy objective remains active throughout training. Training uses bfloat16 precision with FP32 attention softmax and gradient reduction.

A.2 Image-Only Model Setting

We follow exactly the experimental protocol of FlashAR Zhou et al. (2026). These experiments are conducted independently from the U0/ScanAR experiments.

We evaluate LlamaGen Sun et al. (2024) for class-conditional image generation on ImageNet Russakovsky et al. (2015) at 256×256 resolution, and Emu3.5-Image-34B Wang et al. (2024) for text-to-image synthesis at 512×512 resolution.

For LlamaGen, post-training is conducted for 25 epochs on ImageNet with a batch size of 512; training from scratch is conducted for 300 epochs on ImageNet with a batch size of 512. For Emu3.5-Image-34B, we use approximately 80K text-image pairs curated from OpenGPT-4o-Image and ShareGPT-4o-Image Chen et al. (2025a;b), and perform post-training for 20K steps.

The training uses a two-stage schedule. During the first 20% of the training steps, we incorporate a distillation loss. During the second stage, the distillation loss is removed. Both stages use a base learning rate of 2×10^{-5} with cosine decay. The auxiliary-loss coefficient is set to $\lambda_{\text{aux}} = 0.05$. The classifier-free guidance scale is set to 2.0 for LlamaGen and 5.0 for Emu3.5-Image-34B. All experiments are performed on eight NVIDIA H200 GPUs using bfloat16 precision.

We compare against the original raster-scan autoregressive model, parallel autoregressive models trained from scratch, including VAR Tian et al. (2024), PAR Wang et al. (2025b), and NAR He et al. (2025), as well as the post-training adaptation method BlockDiffusion Cui et al. (2025b).

For LlamaGen, we report FID Heusel et al. (2017), Inception Score (IS) Salimans et al. (2016), precision, recall, and sFID. For Emu3.5-Image-34B, we use GenEval Ghosh et al. (2023) to measure compositional fidelity. Inference latency is measured on a single NVIDIA H20-141G GPU with batch size one and averaged over 100 generated samples.

B More Experimental Results

B.1 GPT-5.4 Evaluation Protocol

We use GPT-5.4 to evaluate depth-conditioned multi-view image generation in embodied robot-manipulation scenes. For each evaluation sample, the evaluator receives a textual scene description, a set of depth maps, and the corresponding generated images. The i-th generated image is paired with the i-th depth map. All views depict the same scene from different camera angles.

The depth maps encode normalized inverse depth within each individual view: brighter pixels indicate regions closer to the camera, while darker pixels indicate regions farther away. Because the values represent relative depth rather than absolute scale, depth faithfulness is evaluated independently for each view.

GPT-5.4 assigns a score from 1 to 10 for each of the following six dimensions:

1. **Depth Faithfulness**: whether the generated image follows the spatial layout, depth ordering, occlusion relationships, and geometric structure specified by the corresponding depth map.
2. **Background**: whether background objects, walls, distant furniture, and other far-field elements agree with the scene description, while avoiding hallucinated background content.
3. **Foreground**: whether the robot arm, gripper, manipulated objects, and other near-field elements are present and correctly arranged according to the scene description.
4. **Lighting**: whether illumination direction, intensity, color temperature, shadows, highlights, and reflections are plausible and consistent across views.
5. **Image Quality**: whether the generated images are clear, sharp, realistic, and free from artifacts such as blur, noise, color banding, distorted geometry, unnatural materials, or broken object boundaries.
6. **Multi-View Consistency**: whether objects, poses, colors, textures, lighting, and scene geometry remain consistent across camera views.

The scores follow a common qualitative scale: scores of 9–10 indicate near-perfect results with no noticeable flaws; scores of 7–8 indicate good results with only minor inaccuracies; scores of 5–6 indicate acceptable results with clear but non-critical errors; scores of 3–4 indicate substantial failures; and scores of 1–2 indicate severely degraded or unusable results.

The evaluator applies two special rules. First, if the numbers of depth maps and generated images differ, only the pairs that can be matched in order are evaluated, and the mismatch itself is not penalized. Second, when only one view is available, multi-view consistency is not applicable and receives a score of 10.

The evaluator returns a JSON object containing one score and one textual explanation for each dimension, together with an overall score. The overall score reflects the overall quality of the generated results and is not required to be the strict arithmetic mean of the six dimension scores. The complete evaluation prompt is provided below.

```
You are an expert evaluator for depth-conditioned multi-view image generation in robot
manipulation scenes.

The task being evaluated is: given a set of depth maps (one per camera view) and a
text description of the scene, generate photorealistic images that faithfully follow
the depth layout and match the text description.

You are given two sets of images:

1. {num_views} depth map(s), each representing the normalized inverse depth from a
different camera view of the same scene. In these maps, brighter pixels are closer to
the camera and darker pixels are farther away. The values represent relative depth
within each individual image, not absolute scale.
2. {num_views} generated image(s), each showing the model's output for the
corresponding camera view.

The depth maps and generated images are provided in the same order - the i-th
generated image corresponds to the i-th depth map.

All views together depict the same scene from different camera angles.

---------------------
Evaluation Dimensions
---------------------

Score each dimension from 1 to 10.

General scoring guideline:

9--10: Excellent - the generated image is near-perfect in this dimension. No
noticeable flaws; any imperfection is negligible and would not be caught without
careful side-by-side comparison.
7--8: Good - overall successful with only minor issues. A few small inaccuracies exist
but do not significantly detract from the result. Most viewers would consider it high
quality.
5--6: Acceptable - the general intent is captured but there are clear, easily
noticeable problems. Some elements are wrong, misplaced, or missing, though the image
is still recognizable as an attempt at the target.
```

```
3--4: Poor - significant failures in this dimension. Multiple major errors are
present; the result only partially resembles what was expected.
1--2: Very poor - almost entirely incorrect or broken. The image fails to satisfy the
basic requirements of this dimension.

Evaluate the following dimensions:

1. Depth Faithfulness
Does each generated image faithfully follow the spatial layout defined by its
corresponding depth map?

The depth map is a normalized inverse depth image: brighter = closer, darker =
farther. It only encodes relative depth within a single view and does not represent
absolute scale. Evaluate each view independently - do NOT compare depth consistency
across different views.

Special rule: if the number of depth maps and generated images differ, only evaluate
the pairs that can be matched in order (e.g., if there are 3 depth maps and 2
generated images, only evaluate the first 2 pairs). Do NOT penalize for the mismatch
in count.

Check (per view):
- Whether near and far objects are placed at correct relative distances as indicated
by the depth map
- Whether the depth ordering of objects is preserved (closer objects properly occlude
farther ones)
- Whether the overall 3D spatial structure (surfaces, furniture, workspace layout)
matches the depth information
- Whether depth gradients and transitions (e.g., a table surface receding, walls at
different distances) are correctly reflected
- Whether the robot arm, gripper, and manipulated objects respect the depth
relationships shown in the map

2. Text-Image Alignment - Background / Far Field
Do the generated images accurately reflect the background and distant elements
described in the scene description?

Important: since multiple views together describe one scene, an object mentioned in
the text only needs to appear in at least one view to count as present. Not every view
needs to show every described element.

Check:
- Whether background objects, walls, distant furniture, and far-field elements
mentioned in the text are present in at least one view
- Whether the overall room layout and environmental context match the description
- Whether there are hallucinated far-field objects not described in the text

3. Text-Image Alignment - Foreground / Near Field
Do the generated images accurately reflect the foreground and near-field elements
described in the scene description?

Important: since multiple views together describe one scene, an object mentioned in
the text only needs to appear in at least one view to count as present. Not every view
needs to show every described element.

Check:
- Whether foreground objects, the robot arm, gripper, manipulated objects, and nearby
items mentioned in the text are present in at least one view
- Whether spatial relationships among near-field objects (e.g., "on top of", "next
to", "inside") are correctly rendered where visible
- Whether there are hallucinated near-field objects not described in the text

4. Text-Image Alignment - Lighting
Do the generated images accurately reflect the lighting conditions described or
implied by the scene description?

Check:
- Whether the overall illumination direction, intensity, and color temperature match
the described or expected lighting
- Whether shadows are consistent with the light source direction and scene geometry
- Whether highlights, reflections, and ambient light behave naturally
- Whether lighting is consistent across views (same light source direction and
intensity)

5. Image Quality
Is each generated image visually clear, sharp, realistic, and free of artifacts?

Check:
- Whether there are visible artifacts such as blurring, noise, color banding, or
distortions
- Whether colors and exposure are natural and well-balanced
- Whether there are structural distortions (warped edges, broken geometry, misshapen
surfaces)
```

```
- Whether material appearances (reflection, transparency, roughness) look natural and
realistic
- Whether fine-grained textures are clearly rendered (wood grain, metal sheen, fabric
weave, etc.)
- Whether object edges are crisp and clean without jagged or smeared boundaries

6. Multi-View Consistency
Are the generated images geometrically and visually consistent with each other?

Special rule: if there is only 1 depth map or only 1 generated image, multi-view
consistency is not applicable - score this dimension 10 automatically.

All views depict the same scene from different angles. They should be coherent as a
set.

Check:
- Whether the same object appears at geometrically consistent positions across views
- Whether colors, textures, and lighting conditions are uniform across views
- Whether there are contradictions (e.g., an object present in one view but missing in
another, or appearing with different shapes or colors)
- Whether the robot arm and gripper pose is consistent across all views
- Whether the overall scene layout is coherent when considering all views together

----------------------
Output Requirements
----------------------

Return JSON only.

Do not include any text outside the JSON.

{
  "depth_faithfulness": {"score": 0, "explanation": ""},
  "text_image_alignment_background": {"score": 0, "explanation": ""},
  "text_image_alignment_foreground": {"score": 0, "explanation": ""},
  "text_image_alignment_lighting": {"score": 0, "explanation": ""},
  "image_quality": {"score": 0, "explanation": ""},
  "multi_view_consistency": {"score": 0, "explanation": ""},
  "overall_score": 0
}

The overall_score should reflect the overall quality of the generation considering all
dimensions (not necessarily a strict average).

Below is the text description of the scene, followed by the depth maps:

Instruction:
{instruction}

Here are the depth maps (one per camera view):

<image1><image2><image3>
```

B.2 U0 RESULT

We conducted additional visualization experiments on the U0 model, including image-text interleaved sequence generation, image generation, and image editing. As shown in Figures 4 and 5, U0 retains strong image-text interleaved generation capabilities after ScanAR post-training, can freely switch between modalities during generation, preserves its unified autoregressive capability, and automatically accelerates image generation. Figure 6 demonstrates its text to image generation capability, while Figure 7 shows its image-editing capability. As can be seen, ScanAR accelerates image generation without compromising image quality.

B.3 LLAMAGEN RESULT

We compare ScanAR with LlamaGen, PAR, NAR, VAR, BlockDiffusion, and FlashAR on the ImageNet 256×256 benchmark. The results are reported in Table 7. ScanAR is trained from scratch for 300 epochs and generates each image in 32 decoding steps.

Across all model sizes, ScanAR achieves the highest Inception Score (IS) and throughput among the compared methods. The IS improves from 244.3 to 306.9 as the model scales from B to XXL, indicating that ScanAR benefits effectively from increased model capacity. At the same time, ScanAR reduces the sequential decoding depth from 256 steps for LlamaGen to 32 steps and achieves throughputs of 669.4, 235.2, 113.4, and 65.3 images per second across the four model sizes, respec-

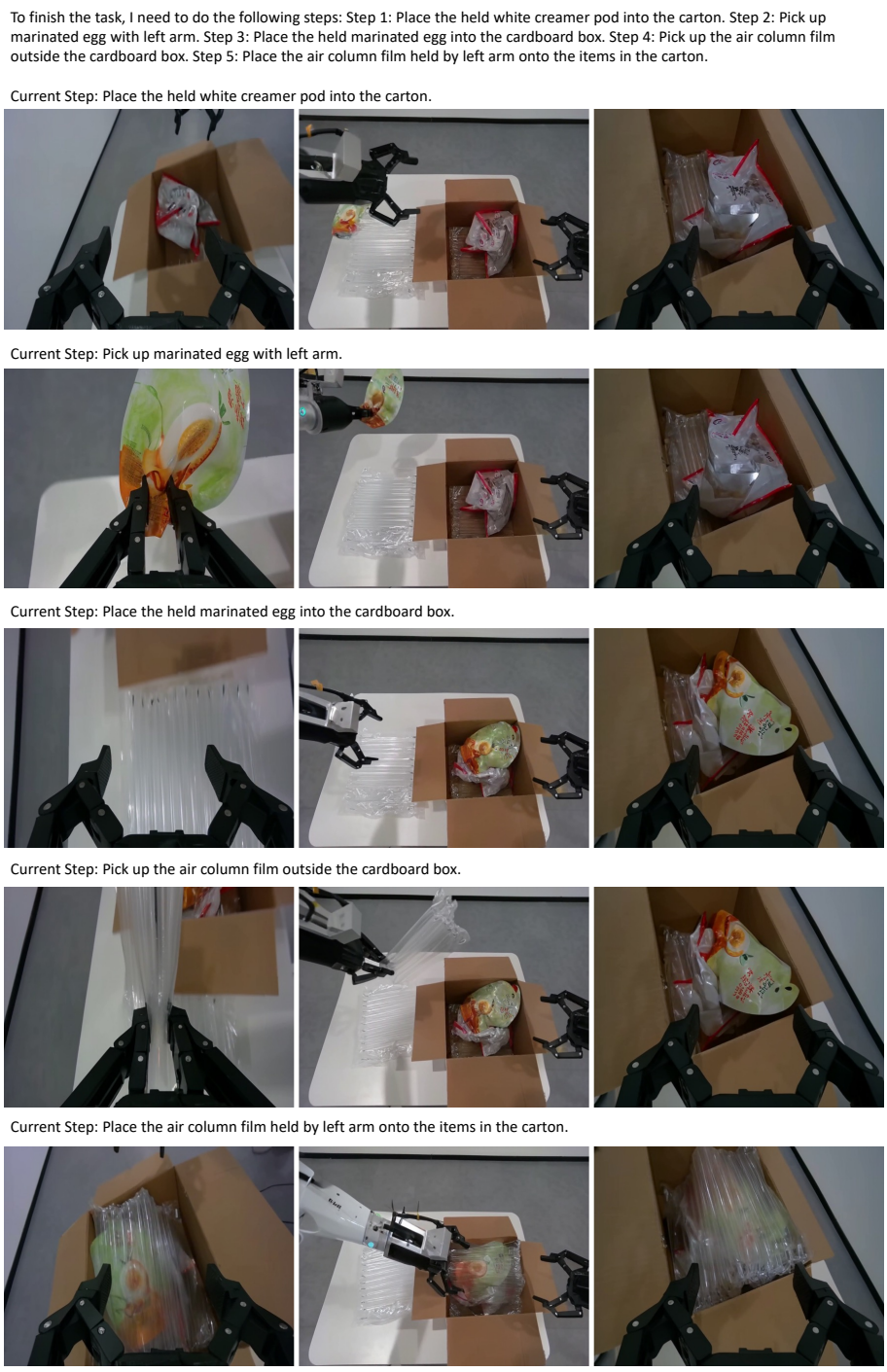


Figure 4: More interleaved generation samples produced by ScanAR posttrained U0.

To finish the task, I need to do the following steps: Step 1: Open the lid of the held canned sugar can with left arm. Step 2: Pick up the spoon from the canned sugar can with left arm. Step 3: Scoop a spoonful of canned sugar with the held spoon by left arm. Step 4: Sprinkle the canned sugar from the spoon into the pot with left arm. Step 5: Place the held spoon into the canned sugar jar with left arm. Step 6: Close the lid of the held canned sugar can with left arm. Step 7: Place the held canned sugar can back on the lower shelf of the double-layer storage rack with right arm. Step 8: Pick up the vinegar from the upper level of the double-layer shelf with right arm.

Current Step: Open the lid of the held canned sugar can with left arm.



Current Step: Pick up the spoon from the canned sugar can with left arm.



Current Step: Scoop a spoonful of canned sugar with the held spoon by left arm.



Current Step: Sprinkle the canned sugar from the spoon into the pot with left arm.



Current Step: Place the held spoon into the canned sugar jar with left arm.



Current Step: Close the lid of the held canned sugar can with left arm.



Current Step: Place the held canned sugar can back on the lower shelf of the double-layer storage rack with right arm.



Current Step: Pick up the vinegar from the upper level of the double-layer shelf with right arm.



Figure 5: More interleaved generation samples produced by ScanAR posttrained U0.



Figure 6: More text to image generation samples produced by ScanAR posttrained U0.



Figure 7: More image edit samples produced by ScanAR posttrained U0.

tively. These results show that ScanAR substantially improves sampling efficiency while preserving strong class-conditional generation quality. The consistent gains across model sizes also demonstrate that the scan-order decoding strategy scales effectively with model capacity.

Qualitative examples of class-conditional image generation by ScanAR-XXL are shown in Figure 8. The generated images demonstrate strong class fidelity across diverse ImageNet categories, while exhibiting diverse appearances and rich visual details.

Table 7: Quantitative evaluation on the ImageNet 256×256 benchmark.

Model size	Method	Type	Epoch	FID↓	IS↑	P/R-F1↑	Steps	Throughput (img/s)
B ($\sim$ 120M)	LlamaGen	From scratch	300	5.46	193.6	0.594	256	117.9
	PAR	From scratch	300	6.21	204.4	0.537	67	174.1
	NAR	From scratch	300	**4.65**	212.3	0.600	31	419.7
	BlockDiffusion	Post-training	75	5.91	176.2	0.589	64	186.3
	FlashAR	Post-training	25	4.68	208.3	**0.605**	31	447.2
	ScanAR	From scratch	300	5.59	**244.3**	0.568	32	**669.4**
L ($\sim$ 360M)	LlamaGen	From scratch	300	3.80	248.3	0.639	256	47.1
	PAR	From scratch	300	4.32	189.4	0.576	67	93.8
	NAR	From scratch	300	**3.06**	263.9	0.641	31	195.4
	VAR	From scratch	200	3.30	274.4	0.634	10	129.3
	BlockDiffusion	Post-training	75	4.55	243.5	0.645	64	103.2
	FlashAR	Post-training	25	3.16	289.0	**0.656**	31	224.7
	ScanAR	From scratch	300	3.86	**295.2**	0.637	32	**235.2**
XL ($\sim$ 700M)	LlamaGen	From scratch	300	3.39	227.1	0.648	256	23.7
	PAR	From scratch	300	3.50	234.4	0.619	67	53.9
	NAR	From scratch	300	**2.70**	277.5	**0.676**	31	98.1
	BlockDiffusion	Post-training	75	4.13	258.6	0.654	64	41.7
	FlashAR	Post-training	25	2.94	293.7	0.672	31	109.3
	ScanAR	From scratch	300	3.34	**301.8**	0.648	32	**113.4**
XXL ($\sim$ 1.4B)	LlamaGen	From scratch	300	3.09	253.6	0.647	256	14.1
	PAR	From scratch	300	3.20	288.3	0.632	67	33.9
	NAR	From scratch	300	**2.58**	293.5	0.673	31	56.9
	BlockDiffusion	Post-training	75	3.78	264.9	0.652	64	26.8
	FlashAR	Post-training	25	2.79	289.4	**0.690**	31	63.4
	ScanAR	From scratch	300	3.46	**306.9**	0.640	32	**65.3**

B.3.1 Throughput Evaluation

We evaluate the inference throughput of Scan-AR on a single NVIDIA H20-3e GPU. All models are evaluated using BF16 arithmetic with PyTorch `2.9.1+cu129`. The autoregressive transformer uses the FlexAttention backend with BlockMask. We generate 256 visual tokens arranged as a 16×16 latent grid, corresponding to 256×256 output images with a VQ downsampling factor of 16.

The sampling configuration is fixed to classifier-free guidance (CFG) scale 2.0, temperature 1.0, disabled top-k filtering ($k = 0$), and top-$p = 1.0$. Random sampling is enabled, and the random seed is set to 0 for reproducibility.

To reduce Python scheduling and CUDA kernel launch overhead, we use a static CUDA Graph execution path. The graph is captured after three warm-up generations, including cache allocation, FlexAttention compilation, and all lazy kernel compilation. The captured graph uses a fixed batch size of $B = 512$. Before each graph replay, the class labels are copied into a preallocated CUDA buffer outside the graph. The graph then performs the complete Scan-AR generation procedure, including the first-row causal decoding, first-row bidirectional commit, subsequent row-wise decoding, and token sampling.

We measure five graph replays per model. Timing is performed with CUDA events and includes only autoregressive token generation. Checkpoint loading, graph capture, warm-up iterations, VQ

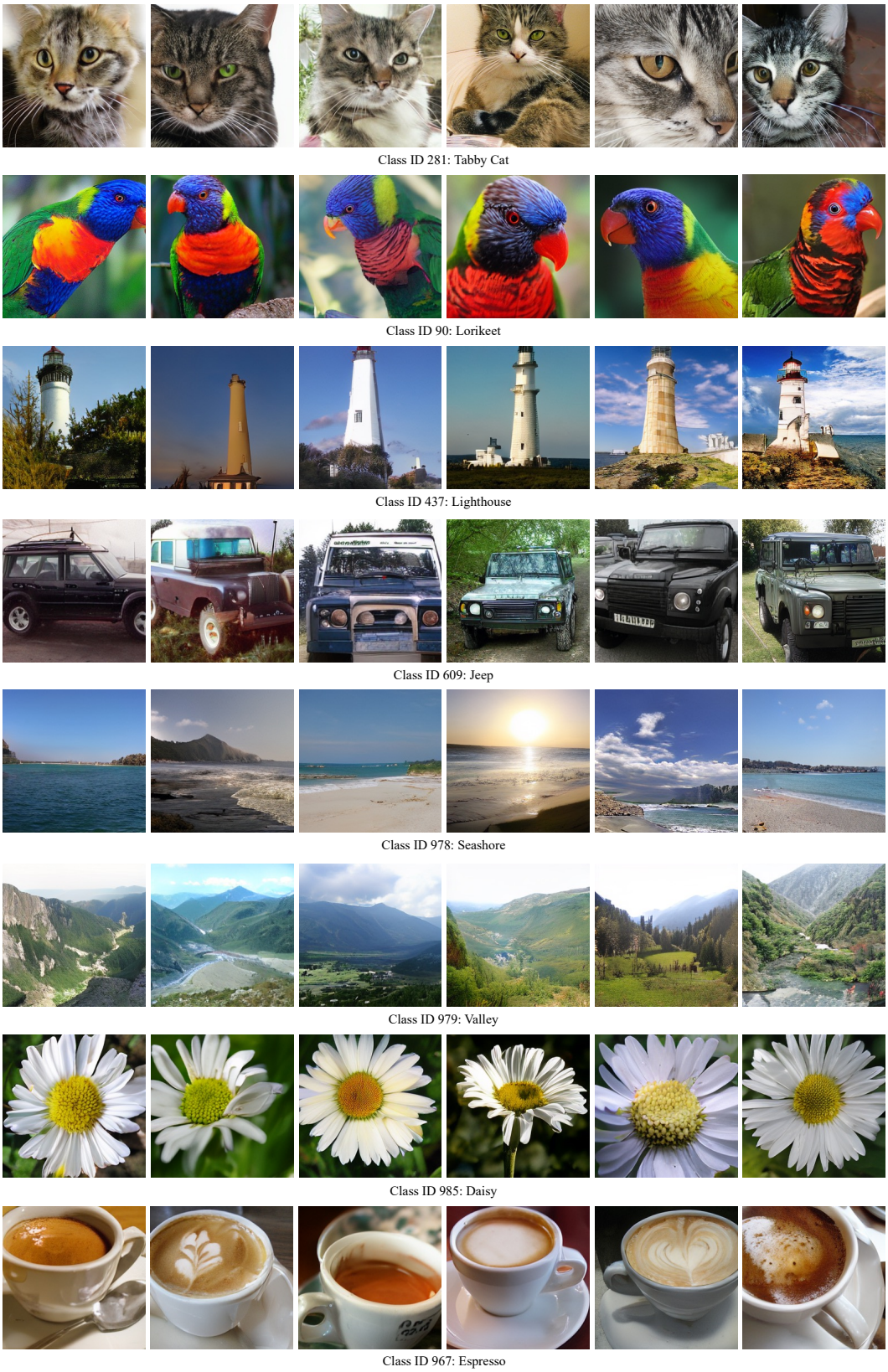


Figure 8: Class-conditional image generation samples produced by ScanAR-XXL on ImageNet 256×256.

Table 8: Single-GPU Scan-AR generation throughput measured with a static CUDA Graph on an NVIDIA H20-3e. All measurements use BF16 precision, batch size 512, a 16×16 visual-token grid, CFG scale 2.0, temperature 1.0, FlexAttention, three warm-up batches, and five timed graph replays. VQ decoding and image I/O are excluded.

Model	Mean batch time (s)	Throughput (img/s)	Peak allocated memory (GiB)
GPT-B	0.764874	669.39	12.14
GPT-L	2.177245	235.16	28.07
GPT-XL	4.516144	113.37	50.52
GPT-XXL	7.840613	65.30	79.58

decoding, image post-processing, and disk I/O are excluded from the timed region. Let t_i denote the elapsed time of the i-th timed batch. We report the generation throughput as

$$\text{Throughput} = \frac{B}{\frac{1}{K}\sum_{i=1}^{K} t_i}, \qquad K = 5, \tag{6}$$

where the unit is generated images per second.

Table 8 reports the measured throughput. Each result is obtained from an independently loaded model on an otherwise idle GPU.

B.4 Additional Results on Emu3.5-Image-34B

B.4.1 Training Cost and Inference Efficiency

Table 9 provides a detailed comparison of training configurations and inference efficiency on Emu3.5-Image-34B. The original Emu3.5-Image model is trained from scratch for 940K steps on 150B training tokens. In contrast, BlockDiffusion, FlashAR, and ScanAR are post-training methods that use 80M training tokens. ScanAR requires only 20K post-training steps, compared with 50K steps for BlockDiffusion and FlashAR, corresponding to a 60% reduction in post-training steps.

Table 9: Training configurations and inference efficiency on Emu3.5-Image-34B.

Method	Type	Steps	Data	Latency (s)↓	Decoding Steps
Emu3.5-Image	From scratch	940K	150B	130.10	1024
BlockDiffusion	Post-training	50K	80M	6.17	64
FlashAR	Post-training	50K	80M	5.68	63
ScanAR	Post-training	20K	80M	4.69	64

The original raster-scan model requires 1024 serial decoding steps for 512×512 image generation, whereas ScanAR requires only 64 steps. This reduces the sequential decoding depth by a factor of 16 and results in a wall-clock latency of 4.69 seconds, compared with 130.10 seconds for the original model. Thus, ScanAR achieves a $27.7\times$ end-to-end speedup. Compared with BlockDiffusion and FlashAR, ScanAR also achieves lower latency, with relative reductions of approximately 24.0% and 17.4%, respectively. The similar decoding-step counts of ScanAR and the other accelerated methods indicate that its latency advantage comes from the efficiency of its row-frontier decoding and post-training design.

C Implementation and Efficiency Details

C.1 Training Implementation and Efficiency

ScanAR augments conventional sequential sampling with parallel row-wise sampling by modifying only the attention mask and loss computation. A practical implementation challenge arises from its block-causal attention pattern. Replacing the standard causal mask prevents direct reuse of the optimized causal-mask execution path in FlashAttention, whereas naively materializing a dense four-dimensional (4-D) mask incurs substantial intermediate-memory and computational overhead.

Table 10: Training efficiency of causal AR and ScanAR with different attention implementations on eight H20E GPUs (141 GB memory, TP$= 4$). Step time is measured after warm-up.

Sequence length	Method	Step time (s)	Peak memory (GB)
4K	Causal AR	10.1	91.9
	ScanAR with dense 4-D mask	16.2	105.6
	ScanAR with FlexAttention	10.5	94.5
8K	Causal AR	20.2	91.9
	ScanAR with dense 4-D mask	–	OOM
	ScanAR with FlexAttention	21.3	94.2

We address this issue using FlexAttention, which represents the ScanAR attention pattern compactly as a `BlockMask`. The fused attention kernel processes the permitted blocks and applies online softmax without materializing the full attention-score matrix or the dense 4-D mask. This representation is well suited to ScanAR because its visible regions are composed primarily of contiguous, aligned row blocks.

Table 10 compares the resulting training efficiency with the original causal AR implementation and a ScanAR implementation using a dense 4-D mask. At a sequence length of 4K, FlexAttention reduces the ScanAR step time from 16.2 to 10.5 seconds and peak memory from 105.6 to 94.5 GB, approaching the causal AR baseline of 10.1 seconds and 91.9 GB. At 8K, the dense-mask implementation runs out of memory, whereas FlexAttention completes a step in 21.3 seconds with 94.2 GB of peak memory, compared with 20.2 seconds and 91.9 GB for causal AR.

The reported peak memory is the maximum allocated memory over the entire training step rather than the memory consumed by the attention kernel alone. In these experiments, both the causal and FlexAttention paths reached their peak allocations during other phases of the training step. Consequently, increasing the sequence length from 4K to 8K does not necessarily increase the measured peak; the small observed differences are also within normal allocator and runtime variation.