MULTISCALE GENERATIVE MODELING BY RECURSIVE REFINEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Physical systems exhibit structure across spatial scales, from large-scale organization to fine details. Coarse representations describe these systems with fewer degrees of freedom but do not uniquely determine their fine-scale structure. Learning the distribution of these unresolved structures allows generation to proceed through smaller refinement problems, each conditioned on the coarse structure already established. Here, we introduce a discrete multiscale generative framework that constructs samples through recursive refinement directly on the native grid, without a learned encoder or decoder. A single model is shared across scales and learns the conditional distribution of fine-scale configurations, with an explicit spatial correspondence between coarse and fine grids. The framework supports different stochastic refinement processes, including spin dynamics for binary structures and mass-conserving dynamics for particle allocation. The conservative processes preserve particle counts exactly, ensuring that every generated refinement aggregates to its prescribed coarse state. We demonstrate the framework on natural images, Julia-sets, and three-dimensional porous media. For Julia-sets, recursive refinement extends to scales unseen during training. For porous media, the model generates volumes of 1024^3 voxels with exactly prescribed porosity (void fraction), with no fixed architectural limit on volume size. We show that a single refinement model can generate structure across scales while honoring coarse-scale constraints.

1 INTRODUCTION

Spatial systems commonly exhibit structure across multiple scales (E, 2011). In porous media, bulk properties such as porosity summarize complex 3D geometries but do not uniquely determine their connectivity nor the spatial distribution of throats that govern fluid transport (Torquato, 2002). In natural images, coarse spatial arrangements establish the scene, while finer details resolve texture and local shape. A coarse description captures useful information but remains compatible with many fine-scale realizations. Generative modeling offers a way to describe this variability: starting from a coarse state, a model can sample finer structures conditioned on the information already available. Repeating this operation builds a hierarchy of increasingly detailed realizations.

Generative models use different strategies to synthesize spatial structure, including direct modeling on the original grid, generation in learned latent spaces, and successive refinement across scales. Direct generation on the native grid becomes increasingly expensive with resolution. Learned compression reduces this cost by operating in a lower-dimensional representation, while coarse-to-fine methods distribute generation across successive refinement stages. In scientific applications, these intermediate coarse representations can also encode physically meaningful quantities that constrain the finer-scale structure. A coarse cell may represent a particle count, a local average, or the volume occupied by a material phase. When such quantities are prescribed, refinement should introduce fine-scale structure while preserving them. We therefore retain intermediate states on the original spatial grid, where prescribed coarse quantities can be enforced directly rather than through conditioning alone.

Here, we introduce *multiscale generative modeling by recursive refinement*, a framework that learns to generate fine-scale structure subject to constraints imposed by an existing coarse state. Each refinement stage acts as a stochastic lifting operation (Gear et al., 2003), and a single neural network

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.

models the refinement dynamics across all spatial scales. The process operates directly on the native grid, without a learned encoder or decoder. Different stochastic refinement processes define the relationship between successive states. In conservative formulations, the network learns how to distribute each coarse cell's among fine cells, while the transition rules enforce exact conservation.

This framework raises two questions: First, can a model trained to refine structures over a finite range of scales learn a rule that remains useful beyond that range? Sharing parameters makes further refinement possible without introducing additional networks, but its statistical validity depends on the structure of the underlying system. Second, can recursive refinement support generation over large spatial domains while retaining exact control over specified quantities? Bounding the spatial context of each refinement allows larger domains to be generated through additional local evaluations, while conservative transitions preserve prescribed quantities.

We investigate these questions using natural images, Julia-sets, and three-dimensional porous media. Natural-image experiments compare refinement processes and demonstrate how a common coarse state can produce distinct fine-scale realizations. Julia-sets test refinement beyond training scales and error accumulation across successive refinements. Porous media test large-volume generation and physical control: we generate 1024^3 voxel geometries with exactly prescribed porosity.

2 Related work

2.1 Learned representations for image generation

Learned representations reduce the cost of image generation by modeling the data distribution in a latent space. VQ-VAE implements this approach by encoding images as grids of discrete codes and learning a prior over them (Van Den Oord et al., 2017). VQGAN builds on this approach with perceptual and adversarial losses that preserve visual detail under stronger compression, allowing a neural network to model the resulting codes for high-resolution generation (Esser et al., 2021). Generation in latent space can then use masked-token prediction, as in MaskGIT (Chang et al., 2022), or diffusion, as in latent diffusion models (Rombach et al., 2022). MAR combines masked autoregressive generation with diffusion to predict continuous tokens conditioned on those already generated (Li et al., 2024). Latent representations can also be organized across scales. VQ-VAE-2 generates finer codes conditioned on coarser ones (Razavi et al., 2019). VAR uses next-scale prediction to progressively add quantized residuals (Tian et al., 2024), while FlexVAR predicts a complete latent representation at each scale, enabling generations for resolutions beyond the ones used during training (Jiao et al., 2026). These approaches rely on a learned decoder to reconstruct the spatial fields. Our framework instead learns refinement directly on the native grid, where intermediate states retain explicit meanings such as particle counts or material occupancy. Therefore, coarse quantities provide interpretable conditions for subsequent generation and, in conservative formulations, are preserved exactly by the refinement dynamics.

2.2 Hierarchical and recursive generation

Coarse-to-fine generation builds spatial structure by progressively adding detail. Early approaches learn successive image refinements through a Laplacian pyramid (Denton et al., 2015), while approaches like SinGAN learn patch distributions at multiple resolutions (Rott Shaham et al., 2019). Diffusion models extend this strategy through stochastic super-resolution conditioned on coarse images, without enforcing exact consistency between resolutions (Saharia et al., 2022; Ho et al., 2022). Wavelet score-based models make the relationship between resolutions explicit by factoring the data distribution into conditional distributions of wavelet coefficients (Guth et al., 2022). The resulting hierarchy improves conditioning and reduces the cost of generation. More recently, renormalization-group flow matching combines coarse-to-fine generation with local computation across successively rescaled lattices (Masuki & Ashida, 2026). Our conservative formulations generate fine structure conditioned on a coarse representation, but operate directly on discrete spatial fields, with transition rules that preserve local particle counts and respect cell capacities.

2.3 GENERATIVE LIFTING AND STRUCTURAL CONSTRAINTS

A coarse description generally admits many fine-scale realizations. In multiscale computation, constructing realizations consistent with that description is known as lifting (Gear et al., 2003). Conditional generative models can learn to sample this unresolved variability, as demonstrated by Crabtree et al. (2023). We adopt this idea for spatial generation, learning a stochastic lifting operation that can be applied recursively across resolutions. Exact consistency with coarse observations can also be built into learned reconstruction. Hard-constrained downscaling enforces conservation explicitly (Harder et al., 2023), while range-null-space decompositions allow generative models to vary fine detail without changing prescribed linear observations (Wang et al., 2022). Our conservative processes enforce consistency through their discrete transition rules, distributing each coarse cell's particle count among its corresponding fine cells.

Discrete diffusion provides the stochastic foundation for our refinement processes. Early approaches developed diffusion on categorical state spaces (Hoogeboom et al., 2021; Austin et al., 2021), while continuous-time formulations describe discrete generation through Markov jump processes (Campbell et al., 2022). Blackout Diffusion develops an exact reverse-time formulation for general discrete-state Markov processes, including the corresponding continuous-time generator, and illustrates the framework using a pure-death process instantiation on discrete data (Santos et al., 2023). The stochastic process can therefore be chosen so that its transition structure reflects properties of the data. Discrete Spatial Diffusion builds on this formulation and introduces spatial particle transport, preserving total intensity exactly throughout the forward and reverse processes (Santos et al., 2026), while CountsDiff extends discrete diffusion to count-valued data (Soatto et al., 2026). Here, we formulate recursive refinement directly in terms of discrete stochastic processes, with each process specifying how a coarse state is lifted to finer structure and which coarse-scale quantities are preserved exactly.

2.4 GENERATION OF SCIENTIFIC SPATIAL FIELDS

Generative models of scientific fields must capture spatial structure while satisfying prescribed physical constraints exactly. In porous media, Mosser et al. (2017) introduced GAN-based reconstruction using training volumes of 64^3 and 128^3, with the computational cost of three-dimensional convolutional networks limiting extension to larger domains. SliceGAN learns three-dimensional microstructures from two-dimensional images (Kench & Cooper, 2021), while latent diffusion enables porosity-conditioned rock generation at 256^3 voxels (Naiff et al., 2025). Beyond porous media, ZoomLDM combines multiscale training with joint sampling to coordinate local detail and global structure in large scientific images (Yellapragada et al., 2025). Discrete spatial diffusion takes a further step toward exact control by conserving total intensity throughout generation, allowing prescribed phase fractions in material microstructures (Santos et al., 2026). This control comes with a demanding sampling procedure: the model operates at full resolution throughout, sampling individual particle transitions with adaptive time stepping. Our framework instead introduces fine cells progressively and confines conservative allocation to local refinement groups. We demonstrate this construction on porous volumes containing 1024^3 voxels with exactly prescribed porosity.

3 MULTISCALE DIFFUSION BY RECURSIVE REFINEMENT

We construct a field by recursively resolving coarse cells into finer cells on the native grid. Within each refinement stage, a forward stochastic process removes fine-scale information while keeping the coarse state fixed. A single neural network, shared across scales, parameterizes the reverse dynamics that generate the newly resolved structure. The choice of stochastic process specifies the relationship between coarse and fine states, including which quantities, if any, are preserved exactly.

3.1 MATHEMATICAL FORMULATION

Let $x^{(0)}, \ldots, x^{(S)}$ denote representations of a sample on progressively finer grids, where S is the number of refinement scales. Starting from a prescribed or sampled coarse representation $x^{(0)}$, we

generate each finer representation conditioned on the coarser one:

$$p_\theta\left(x^{(1)}, \ldots, x^{(S)} \mid x^{(0)}\right) = \prod_{s=0}^{S-1} p_\theta\left(x^{(s+1)} \mid x^{(s)}, s\right). \tag{1}$$

Each conditional distribution is generated by reversing a stochastic process, with the same network parameters θ shared across all refinement stages. Conditioning on the scale index s is optional.

Within each refinement stage, the coarse representation $c = x^{(s)}$ remains fixed. Let Z_t denote the evolving discrete configuration, including any reservoir variables or flip indicators. Forward time increases as fine-scale information is removed. Generation begins from the endpoint of the forward process and evolves backward in time, while the coarse state c remains fixed.

Let $r_t(z, z' \mid c)$ denote the rate of a forward transition from z to z', and let $p_t(z \mid c)$ be the probability of configuration z at time t, given the coarse representation c. Reversing the process reverses each allowed transition and adjusts its rate according to the relative probabilities of the two configurations (Santos et al., 2023):

$$\bar{r}_t(z, z' \mid c) = r_t(z', z \mid c)\frac{p_t(z' \mid c)}{p_t(z \mid c)}, \qquad z' \neq z, \tag{2}$$

for $p_t(z \mid c) > 0$. The network parameterizes an approximation to these reverse rates, while the chosen process determines which transitions are allowed.

During training, we sample a refinement stage and a corresponding coarse-fine pair, then evolve the fine representation forward in time while holding the coarse representation fixed. Because the original sample is known during training, we can compute target reverse rates $\widetilde{r}_t$ using the formula above, with probabilities conditioned on that sample as well as c. The network predicts the quantities needed to construct the reverse rates $\bar{r}_\theta$ from the evolving fine state, coarse input, and process time.

For a time grid $0 = t_0 < \cdots < t_K = T$, a discrete-time approximation to the reverse-rate objective is:

$$\mathcal{L}(\theta) = \mathbb{E}\left[w_k \sum_{\text{allowed } z'} (\bar{r}_\theta - \widetilde{r}_{t_k} \log \bar{r}_\theta)\right], \qquad w_k = t_k - t_{k-1}. \tag{3}$$

The expectation is over training samples, refinement stages, forward-process states, and uniformly sampled indices $k \in \{1, \ldots, K\}$. Both rates are evaluated for the transition $Z_{t_k} \to z'$, and w_k is the corresponding time interval. This objective follows the transition-rate derivation in Santos et al. (2023), applied to spatial diffusion in Santos et al. (2026).

Each stochastic process defines which changes are allowed during refinement. In reservoir allocation, the forward process independently transfers particles from fine cells to their corresponding coarse cell reservoirs. The number remaining in fine cell j therefore follows $N_{t,j} \mid N_{0,j} \sim \mathrm{Binomial}(N_{0,j}, \alpha(t))$, where $\alpha(t)$ is the survival probability. Spin refinement applies the same removal process to the binary flips that distinguish the fine sample from nearest-neighbor upsampling of the coarse input. Local redistribution moves particles between fine cells within the same refinement group, following the spatial-jump construction of Santos et al. (2026).

For conservative processes, let $\mathcal{F}_s(i)$ denote the set of fine cells contained within coarse cell i at refinement stage s. Both forward and reverse transitions preserve:

$$R_{t,i} + \sum_{j \in \mathcal{F}_s(i)} N_{t,j} = c_i, \tag{4}$$

where $R_{t,i}$ is the reservoir count and c_i is the prescribed coarse count. Processes without a reservoir have $R_{t,i} = 0$ throughout. For reservoir allocation, all particles must be transferred from the reservoir to the fine cells before the next refinement stage begins, so the generated fine counts sum to their prescribed coarse counts. Repeating this construction preserves count consistency across the hierarchy. The guarantee follows from the allowed transitions, independently of the accuracy of the learned rates. Spin refinement does not impose count conservation. Axis-wise allocation applies this construction to two fine cells at a time, alternating between spatial axes. These processes are illustrated in Figure 1.

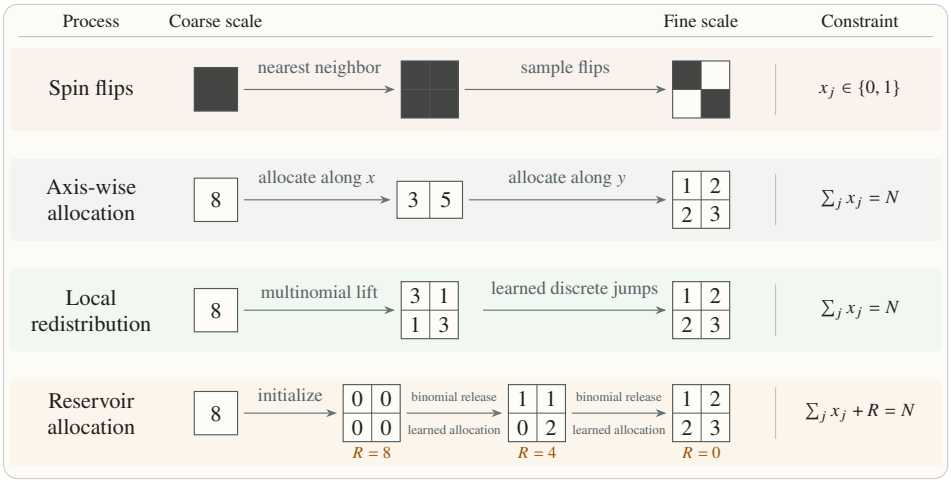

Figure 1: **Discrete stochastic refinement processes.** Spin flips modify a nearest-neighbor upsampling of the coarse state while retaining binary values. Axis-wise allocation distributes particles along one spatial axis at a time, alternating between axes. Local redistribution starts from a multinomial allocation and transfers particles between adjacent fine cells within each 2×2 group. Reservoir allocation progressively releases particles into initially empty fine cells through binomial release and learned allocation. The three allocation processes conserve the prescribed coarse count N exactly: the sum of the fine-cell counts x_j and any remaining reservoir R equals N. Here, $N = 8$.

3.2 Shared refinement model

At each refinement stage, the network receives one token per coarse cell. Each token contains the coarse-cell value, the current state of its corresponding fine cells (four in 2D and eight in 3D for the regular $2\times$ refinement used here), normalized spatial coordinates, and the local process time t. We additionally provide the scale index when the refinement statistics are explicitly scale dependent. For particle-allocation processes, the token also contains the number of particles remaining in the coarse-cell reservoir. For spin refinement, the fine state is represented by the flip indicators relative to nearest-neighbor upsampling of the coarse state. We use a transformer encoder to predict the parameters of the reverse transitions. Its parameters are shared across all refinement stages. In our experiments, the transformer model is particularly convenient because it operates on a variable number of spatial tokens, allowing the same network to process grids of different sizes. The refinement framework itself does not depend on this choice of architecture.

For sufficiently small grids, all coarse cells are processed jointly. For larger grids, specifically the three-dimensional domains, we restrict each evaluation to a spatial neighborhood containing at most a fixed number of tokens. Each neighborhood includes multiple coarse cells together with their associated fine-cell groups, allowing refinement to account for interactions across neighboring coarse cells. At coarser scales, the same number of tokens spans a larger physical region of the sample. The hierarchy therefore captures long-range structure at coarse resolutions and progressively resolves more local structure at finer resolutions.

Training uses coarse-fine pairs obtained from the data hierarchy. We sample a refinement stage s and process time t, apply the corresponding forward dynamics to the fine state while keeping the coarse state fixed, and train the network to predict the quantities required by the reverse process. The stochastic process determines the admissible transitions and their constraints (the network only parameterizes their rates). Therefore, conservation in the particle-allocation processes follows from the transition rules and does not depend on the accuracy or architecture of the neural network. In this work, all experiments use regular lattices. The construction itself only requires a correspondence between coarse and fine cells together with a definition of the admissible local transitions, and is therefore not restricted to Cartesian grids.

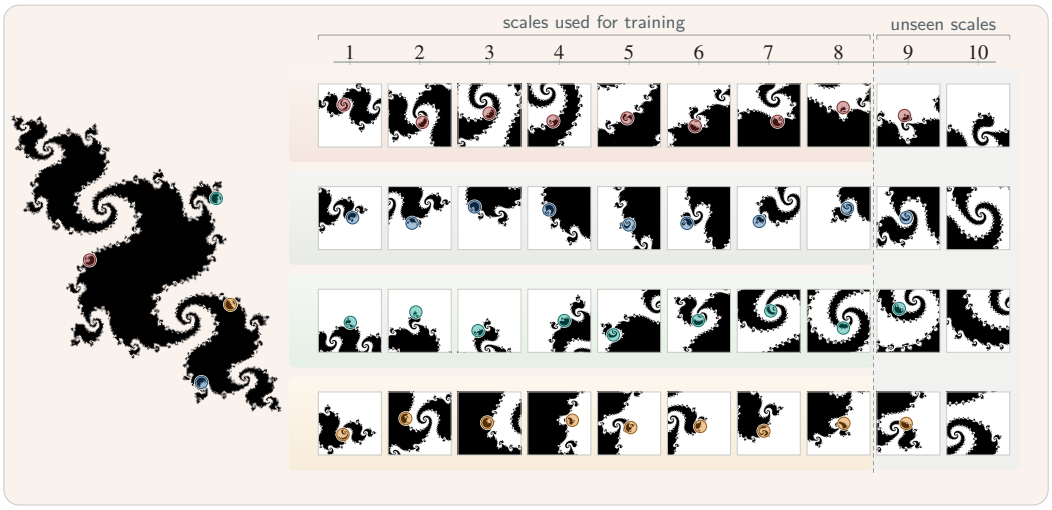

Figure 2: **Julia-set trajectories across successive spatial scales.** Scale 0 (left) shows the initial domain, rotated by 45° for layout. Colored markers identify the starting locations of four zoom autoregressive trajectories. The corresponding trajectories are shown as rows on the right. Each column represents a successive $2\times$ zoom, rendered on a 512×512 grid, and the marker within each frame indicates the center of the next refinement. Scales 1-8 are used during training, while scales 9-10 are held out for evaluation.These trajectories are used to evaluate whether a shared refinement model without explicit scale conditioning extrapolates beyond the spatial scales observed during training.

4 Fractal scale extrapolation

We use a single Julia-set as a proof of concept to test whether a refinement rule learned over a finite range of scales remains useful at finer resolutions. Each refinement begins from nearest-neighbor upsampling of the coarse binary state. Spin dynamics represent the fine structure through binary flips relative to this upsampled state. In the forward process these flips are progressively removed, while the learned reverse process restores them using probabilities obtained from the network outputs. The same model is used at every scale, with no explicit scale conditioning.

We train on scales 1-8 and evaluate autoregressively through scale 10, so that the final two refinements are outside the training range (as seen in Figure 2). The spatial area represented in the training data also decreases rapidly with scale, providing progressively less data coverage for the finer refinements. At inference, each predicted state becomes the coarse input to the next stage, causing errors to accumulate across the full trajectory.

Appendix B compares this recursive rollout with nearest-neighbor refinement and an equivalent transformer trained directly on coarse-fine image pairs using binary cross-entropy. The deterministic model progressively accumulates error, while recursive nearest-neighbor interpolation fails to recover the unresolved boundary structure. In contrast, our refinement model remains accurate throughout the training scales and continues to produce low-error refinements at the two unseen scales. At scale 10, the relative pixel errors are 3%, 18%, and 23% for our model, the deterministic model, and recursive nearest neighbor, respectively. This scale extrapolation is not restricted to fractals or binary data. In Appendix G, we apply the same idea to natural images and recursively refine CIFAR-10 samples beyond the resolutions encountered during training.

5 Conservative generation of natural images

We evaluate recursive refinement on natural images using conservative particle-allocation processes (depicted in Figure 3). Each color channel is represented as a count field, and the fine-cell counts are constrained to aggregate exactly to their prescribed coarse-cell values. In contrast to the fractal experiments, we provide the scale index explicitly to the network. The resulting construction is loosely analogous to coarse-to-fine processing in the visual cortex, where coarse spatial information

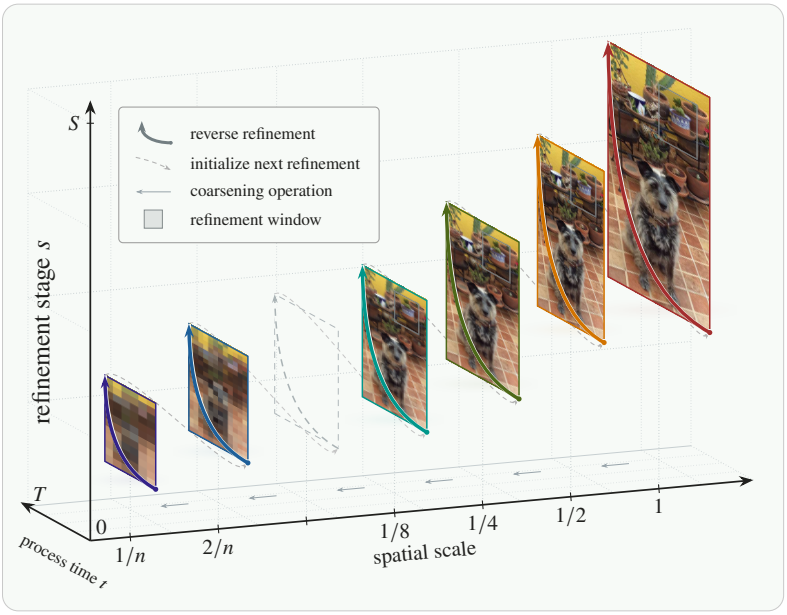

Figure 3: **Schematic of multiscale generation by recursive refinement.** Generation proceeds from coarse to fine through successive stochastic refinement stages. Within each stage, generation evolves backward in process time from T to 0, producing fine structure conditioned on the fixed coarse state (solid curves). The resulting fine configuration becomes the coarse state for the next refinement stage, where process time resets (dashed arrows). Gray arrows show the corresponding forward coarsening direction. A single model parameterizes the refinement dynamics across scales, and the shaded region denotes the spatial refinement window.

Table 1: Best CIFAR-10 FID obtained by each allocation mechanism trained for 2M steps. We used 10,000 generated samples with the same sampling effort and computed FID against the test set. Lower is better. $(^*)$ For local redistribution, the online weights outperform the EMA weights and are therefore reported.

Method	Sampling weights	10k FID ↓
Local redistribution	Online*	40.1
Axis-wise allocation	EMA	26.6
Reservoir allocation	EMA	21.1

is resolved before finer detail (Schyns & Oliva, 1994). We explore three refinement processes. In reservoir allocation, each coarse cell is refined jointly into a 2×2 group of four fine cells. In axis-wise allocation, each coarse cell is split into two cells along one spatial axis at a time, alternating between axes while holding the other fixed, in analogy with coordinate-wise Gibbs updates (Geman & Geman, 1984). We also consider local redistribution, in which particles move among the four fine cells within each 2×2 group. The three processes are illustrated in Figure 1.

We compare these three processes on CIFAR-10 using the same training and sampling budgets. All models are trained on a single HP ZGX nano workstation using AdamW with a learning rate of 10^{-4}, without sweeping the optimizer or learning rate. Table 1 summarizes the results. All three processes generate coherent samples, and we do not interpret this comparison as establishing one refinement process as generally superior to the others. Under the settings tested here, reservoir allocation achieves the lowest FID, reaching 21.1 on 10,000 generated samples. These experiments are not intended to compete with state-of-the-art image generation models. A more relevant reference is Discrete Spatial Diffusion, which also enforces exact conservation during generation: our best model improves its reported FID of 46.3 while requiring substantially fewer fine-grid updates during sampling. Given the small model, modest hardware, and absence of systematic hyperparameter tuning, we treat these results as a proof of concept rather than a performance limit. Larger models and longer training remain unexplored. Detailed ablations and sampling-cost comparisons are provided in Appendix C.

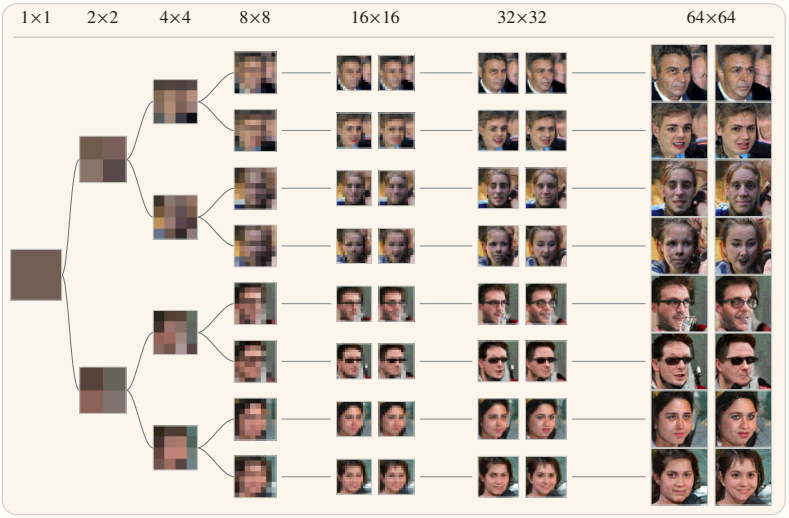

Figure 4: **A family of images generated by recursive refinement.** Starting from a common 1×1 coarse state, trajectories branch into different realizations and progress to 64×64 images. Each branching point samples alternative refinements conditioned on the same coarse configuration. Branching at coarse scales produces substantial differences in appearance and spatial organization, while later branching introduces finer variations within the structure already established.

Finally, we apply reservoir allocation to FFHQ. The branching trajectories in Figure 4 illustrate how variability changes across the hierarchy. Branches introduced at coarse scales can lead to substantially different faces, whereas branches introduced at later stages mainly affect finer features. Because each refinement must remain consistent with its coarse state, the variability introduced at a given stage is restricted to structure that has not yet been resolved. Additional results and quantitative details are provided in Appendix E.

6 Large-scale porous-media generation

We next use three-dimensional porous media to test whether recursive refinement can generate volumes substantially larger than those used in conventional full-grid generative models (Sadeghkhani et al., 2026). By construction, the framework imposes no fixed upper limit on the spatial size of generated samples while preserving prescribed coarse-scale quantities exactly. To our knowledge, this is the first generative framework to combine generation at arbitrarily large three-dimensional spatial resolutions with exact preservation of prescribed coarse-scale quantities.

To achieve this, we limit each network evaluation to a fixed number of spatial tokens, as described by Algorithm 1, and apply the model locally throughout the domain. Larger volumes require more network evaluations, rather than larger network inputs. At each refinement stage, the model predicts how the mass in each coarse region is distributed among its finer cells, as illustrated in Figure 5.

The final binary representation provides an additional computational advantage. If a coarse cell contains only solid voxels, its refinement is uniquely determined by conservation and no network evaluation is required. Only mixed cells, for which multiple fine-scale configurations are compatible with the coarse count, need to be processed. As refinement proceeds, an increasing fraction of the domain becomes fully determined in this way. Figure 5 shows that by the 128^3 level, large regions can already be skipped, while predictions for the remaining mixed cells can be evaluated in batches.

We use the same porous media samples (Neumann et al., 2021) considered in Discrete Spatial Diffusion (DSD) (Santos et al., 2026), but retain the complete 1000^3 volumes rather than extracting 256^3 training windows. We zero-pad each volume to 1024^3 to construct the refinement hierarchy. A direct extension of DSD to this resolution would be computationally prohibitive. Under no-flux boundary conditions, DSD requires transition probabilities for the spatial random walk between all voxel locations at 1024^3 resolution, the underlying Markov process contains more than 10^9 spatial states, with the transition representation growing linearly with the number of observation times. Their periodic

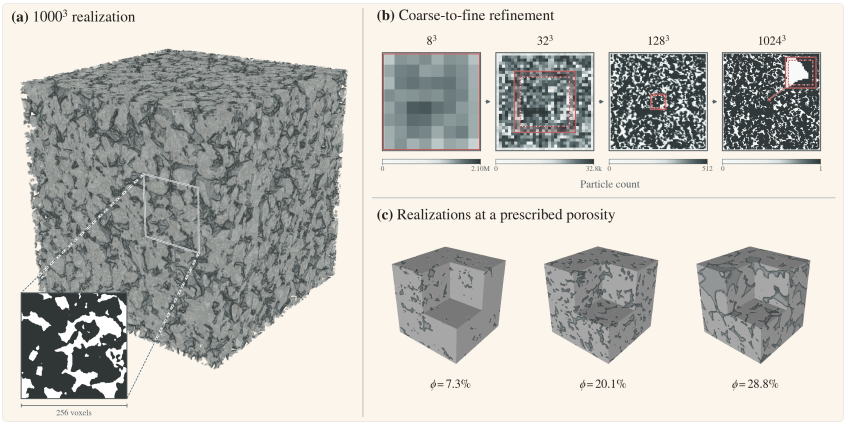

Figure 5: **Recursive generation of three-dimensional porous media.** (a) A 1000^3 voxel realization of Bentheimer sandstone. (b) Cross-sections at selected refinement levels, from 8^3 to 1024^3 voxels, showing the emergence of fine-scale pore structure. The red boxes show the region processed at each refinement step: an 8^3 central window surrounded by a one-cell halo, with only the central region being updated. (c) Realizations at prescribed porosities 7.3%, 20.1%, 28.8%. Conservative refinement preserves the particle count across scales. The physical samples contain 1000^3 voxels and are zero-padded to 1024^3 to allow the power-of-two refinement hierarchy.

formulation is substantially less memory intensive because the transition probabilities can be computed in Fourier space, but it introduces wrap-around transport between opposite boundaries. This is inappropriate for the present samples, which represent finite subvolumes of a larger rock rather than periodic domains, and would introduce artificial connectivity for flow simulations.

Figure 5 shows recursive refinement from coarse count fields to 1024^3 binary samples. Because particle allocation is conservative at every refinement stage, the final porosity is determined exactly by the initial $1 \times 1 \times 1$ state. We generate samples at prescribed porosities while preserving pore morphology consistent with the training data. This experiment targets large-volume generation and exact porosity control rather than comparison with existing 256^3 generators. The open-source dataset contains eleven 1000^3 volumes drawn from distinct rock samples and domains, so a conventional held-out evaluation would leave too little data for a meaningful distributional assessment. We focus on the properties this experiment is designed to test directly: generation of billion-voxel-scale microstructures and exact control of porosity. These experiments condition generation on a prescribed porosity. Unconditional generation could, in principle, be formulated with a nonconservative process such as the spin dynamics used for the Julia-set experiments, but we do not explore that setting here.

7 Conclusion

We introduce recursive refinement as a generative framework for constructing spatial fields across scales. We decompose generation into a sequence of stochastic lifting operations conditioned on progressively finer coarse states. A single model is reused throughout the hierarchy, while the choice of stochastic process determines the admissible fine-scale transitions. For conservative processes, this separates learning from constraint enforcement: the network learns where fine structure should appear, while the transition rules guarantee exact consistency with the prescribed coarse state.

This construction changes how generative modeling scales with resolution, larger fields do not require a larger output layer or a learned latent representation, they require additional applications of the same local refinement rule. We demonstrated this on Julia sets, where the learned rule remains useful beyond the scales observed during training, and on natural images, where successive refinements introduce stochastic detail while exactly preserving their coarse representations. For porous media, the same framework generates binary volumes of 1024^3 voxels with prescribed porosity, while evaluating only the regions whose fine-scale structure remains unresolved. Our results suggest that multiscale structure and physical constraints need not be imposed after generation but they can be built into the generative process itself.

REFERENCES

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11305–11315. IEEE, 2022.

Ellis R Crabtree, Juan M Bello-Rivas, Andrew L Ferguson, and Ioannis G Kevrekidis. Gans and closures: Micro-macro consistency in multiscale modeling. *Multiscale modeling & simulation*, 21(3):1122–1146, 2023.

DeepLearning.AI. Image resolution in black and white. `https://www.deeplearning.ai/the-batch/image-resolution-in-black-and-white`, July 2020. The Batch, July 1, 2020. Accessed September 22, 2026.

Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. *Advances in neural information processing systems*, 28, 2015.

Weinan E. *Principles of Multiscale Modeling*. Cambridge University Press, 2011.

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

C William Gear, James M Hyman, Panagiotis G Kevrekidid, Ioannis G Kevrekidis, Olof Runborg, and Constantinos Theodoropoulos. Equation-free, coarse-grained multiscale computation: Enabling mocroscopic simulators to perform system-level analysis. 2003.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6): 721–741, 1984.

Florentin Guth, Simon Coste, Valentin De Bortoli, and Stephane Mallat. Wavelet score-based generative modeling. *Advances in neural information processing systems*, 35:478–491, 2022.

Paula Harder, Alex Hernandez-Garcia, Venkatesh Ramesh, Qidong Yang, Prasanna Sattegeri, Daniela Szwarcman, Campbell Watson, and David Rolnick. Hard-constrained deep learning for climate downscaling. *Journal of Machine Learning Research*, 24(365):1–40, 2023.

Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33, 2022.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems*, 34:12454–12465, 2021.

Siyu Jiao, Gengwei Zhang, Yinlong Qian, Jiancheng Huang, Yao Zhao, Humphrey Shi, Lin Ma, Yunchao Wei, and Zequn Jie. Flexvar: Flexible visual autoregressive modeling without residual prediction. *Advances in Neural Information Processing Systems*, 38:8955–8978, 2026.

Steve Kench and Samuel J Cooper. Generating three-dimensional structures from a two-dimensional slice with generative adversarial network-based dimensionality expansion. *Nature Machine Intelligence*, 3(4):299–305, 2021.

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.

Kanta Masuki and Yuto Ashida. Renormalization group flow matching for scalable local generative modeling. *arXiv preprint arXiv:2608.23696*, 2026.

Lukas Mosser, Olivier Dubrule, and Martin J Blunt. Reconstruction of three-dimensional porous media using generative adversarial neural networks. *Physical Review E*, 96(4):043309, 2017.

Danilo Naiff, Bernardo P Schaeffer, Gustavo Pires, Dragan Stojkovic, Thomas Rapstine, and Fabio Ramos. Controlled latent diffusion models for 3d porous media reconstruction. *Computers & Geosciences*, pp. 106038, 2025.

Rodrigo F Neumann, Mariane Barsi-Andreeta, Everton Lucas-Oliveira, Hugo Barbalho, Willian A Trevizan, Tito J Bonagamba, and Mathias B Steiner. High accuracy capillary network representation in digital rock reveals permeability scaling functions. *Scientific reports*, 11(1):11370, 2021.

Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. ieee, 2022.

Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. SinGAN: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4569–4579, 2019. doi: 10.1109/ICCV.2019.00467.

Ali Sadeghkhani, Brandon Bennett, Masoud Babaei, and Arash Rabbani. A decade of generative adversarial networks for porous material reconstruction. *arXiv preprint arXiv:2603.11836*, 2026.

Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE transactions on pattern analysis and machine intelligence*, 45(4):4713–4726, 2022.

Javier E Santos, Zachary R Fox, Nicholas Lubbers, and Yen Ting Lin. Blackout diffusion: generative diffusion models in discrete-state spaces. In *International Conference on Machine Learning*, pp. 9034–9059. PMLR, 2023.

Javier E Santos, Agnese Marcato, Roman Colman, Nicholas Lubbers, and Yen Ting Lin. Discrete spatial diffusion: Intensity-preserving diffusion modeling. *Advances in Neural Information Processing Systems*, 38:126425–126458, 2026.

Philippe G. Schyns and Aude Oliva. From blobs to boundary edges: Evidence for time- and spatial-scale-dependent scene recognition. *Psychological Science*, 5(4):195–200, 1994.

Renzo G Soatto, Anders Hoel, Greycen Ren, Shorna Alam, Stephen Bates, Nikolaos P Daskalakis, Caroline Uhler, and Maria Skoularidou. Countsdiff: A diffusion model on the natural numbers for generation and imputation of count-based data. *arXiv preprint arXiv:2604.03779*, 2026.

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.

Salvatore Torquato. *Random Heterogeneous Materials: Microstructure and Macroscopic Properties*, volume 16 of *Interdisciplinary Applied Mathematics*. Springer, New York, 2002. doi: 10.1007/978-1-4757-6355-3.

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

Yinhuai Wang, Jiwen Yu, and Jian Zhang. Zero-shot image restoration using denoising diffusion null-space model. *arXiv preprint arXiv:2212.00490*, 2022.

Srikar Yellapragada, Alexandros Graikos, Kostas Triaridis, Prateek Prasanna, Rajarsi Gupta, Joel Saltz, and Dimitris Samaras. Zoomldm: Latent diffusion model for multi-scale image generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23453–23463. IEEE, 2025.

A Training and sampling algorithms

Algorithm 1 describes training for the reservoir-based particle process. Each iteration samples a spatial scale and process time, removes particles from the fine state, and records them in the corresponding coarse reservoirs. The network learns to predict the missing counts from the partially depleted fine state and coarse conditioning. For large grids, training uses spatially aligned local neighborhoods.

Algorithm 2 describes recursive generation from a prescribed initial mass. At each scale, particles are progressively released from the coarse reservoirs into the fine cells. Birth proposals are limited by both the available reservoir and fine-cell capacities, preserving the combined fine-state and reservoir count at every step. The resulting fine state supplies the coarse input for the next stage, with the same network reused across scales.

Algorithm 1 Training the reservoir-allocation model

Require: Training set $\mathcal{D}$, number of discrete-time observations K, noise schedule $\{\alpha_k\}_{k=0}^{K}$, network f_θ, and refinement-stage capacities $\{C_s\}_{s=0}^{S-1}$

repeat
  Draw a minibatch of finest-scale samples $\mathbf{x}^{(S)} \sim \mathcal{D}$
  Draw a refinement stage $s \sim \mathrm{Uniform}\{0, \dots, S-1\}$
  $\mathbf{N}_0 \leftarrow \mathrm{Coarsen}^{S-s-1}(\mathbf{x}^{(S)})$ ▷ fine counts at the selected stage
  $c_i \leftarrow \sum_{j \in \mathcal{F}_s(i)} N_{0,j}$ for all coarse cells i ▷ fixed coarse state
  Draw $k \sim \mathrm{Uniform}\{1, \dots, K\}$ independently for each sample
  $\mathbf{N}_k \sim \mathrm{Binomial}(\mathbf{N}_0, \alpha_k)$ ▷ element-wise
  $\mathbf{m}_k \leftarrow \mathbf{N}_0 - \mathbf{N}_k$ ▷ removed particles
  $R_{k,i} \leftarrow \sum_{j \in \mathcal{F}_s(i)} m_{k,j}$ for all coarse cells i ▷ reservoir counts
  if $\mathbf{c}$ contains more than $1,000$ coarse-cells **then**
    Draw a local neighborhood $\mathcal{N}$ of $1,000$ coarse-cell tokens
    $(\mathbf{N}_k, \mathbf{c}, \mathbf{R}_k, \mathbf{m}_k) \leftarrow \mathrm{Crop}_{\mathcal{N}}(\mathbf{N}_k, \mathbf{c}, \mathbf{R}_k, \mathbf{m}_k)$ ▷ retain spatially aligned tensors
  end if
  $\boldsymbol{\lambda}_\theta \leftarrow \mathrm{Softplus}(f_\theta(\mathbf{N}_k, \mathbf{c}, \mathbf{R}_k, \alpha_k, s))$
  $\omega_k \leftarrow \alpha_{k-1} - \alpha_k$
  $\mathcal{L} \leftarrow \omega_k \ \mathrm{Mean}\left[\boldsymbol{\lambda}_\theta - \dfrac{\mathbf{m}_k}{C_s} \log(\boldsymbol{\lambda}_\theta)\right]$
  Update θ using $\nabla_\theta \mathcal{L}$
until convergence

Algorithm 2 Multiscale reservoir-allocation sampling

Require: Initial coarse state $\mathbf{x}^{(0)}$, trained network f_θ, numbers of reverse-time observations $\{K_s\}_{s=0}^{S-1}$, and refinement-stage capacities $\{C_s\}_{s=0}^{S-1}$

$\mathbf{c} \leftarrow \mathbf{x}^{(0)}$
for $s = 0, \dots, S-1$ **do**
  $\alpha_k^{(s)} \leftarrow \cos^2\left(\frac{\pi k}{2K_s}\right)$ for $k = 0, \dots, K_s$ ▷ noise schedule
  $\mathbf{N}_{K_s} \leftarrow \mathbf{0}$ ▷ initialize fine cells
  $\mathbf{R}_{K_s} \leftarrow \mathbf{c}$ ▷ initialize coarse reservoirs
  for $k = K_s, \dots, 1$ **do**
    $\boldsymbol{\lambda}_\theta \leftarrow \mathrm{Softplus}\Big(f_\theta(\mathbf{N}_k, \mathbf{c}, \mathbf{R}_k, \alpha_k^{(s)}, s)\Big)$
    $\mathbf{a}_k \leftarrow C_s - \mathbf{N}_k$ ▷ available fine-cell capacity
    if $k = 1$ **then** ▷ release all remaining particles
      for all coarse cells i **do**
        $\mathbf{b}_{1,\mathcal{F}_s(i)} \leftarrow \mathbf{0}$
        for $q = 1, \dots, R_{1,i}$ **do**
          Sample $j \in \mathcal{F}_s(i)$ with probability

$$p_j \propto \lambda_{\theta,j}\, \mathbb{I}[N_{1,j} + b_{1,j} < C_s]$$

          $b_{1,j} \leftarrow b_{1,j} + 1$
        end for
      end for
    else
      $\beta_k^{(s)} \leftarrow \dfrac{\alpha_{k-1}^{(s)} - \alpha_k^{(s)}}{1 - \alpha_k^{(s)}}$
      $\widehat{\mathbf{m}}_k \leftarrow \left\lfloor C_s \boldsymbol{\lambda}_\theta + \frac{1}{2} \right\rfloor$ ▷ predicted missing particles
      $\widehat{\mathbf{m}}_k \leftarrow \max(\mathbf{0}, \min(\widehat{\mathbf{m}}_k, \mathbf{a}_k))$
      $\widetilde{\mathbf{b}}_k \sim \mathrm{Binomial}(\widehat{\mathbf{m}}_k, \beta_k^{(s)})$ ▷ element-wise birth proposals
      for all coarse cells i **do**
        if $\sum_{j \in \mathcal{F}_s(i)} \widetilde{b}_{k,j} > R_{k,i}$ **then**
          $\mathbf{b}_{k,\mathcal{F}_s(i)} \sim \mathrm{MultiHypergeometric}\Big(\widetilde{\mathbf{b}}_{k,\mathcal{F}_s(i)}, R_{k,i}\Big)$ ▷ retain $R_{k,i}$ proposals
        else
          $\mathbf{b}_{k,\mathcal{F}_s(i)} \leftarrow \widetilde{\mathbf{b}}_{k,\mathcal{F}_s(i)}$
        end if
      end for
    end if
    $\mathbf{N}_{k-1} \leftarrow \mathbf{N}_k + \mathbf{b}_k$
    $R_{k-1,i} \leftarrow R_{k,i} - \sum_{j \in \mathcal{F}_s(i)} b_{k,j}$ for all coarse cells i
  end for
  $\mathbf{x}^{(s+1)} \leftarrow \mathbf{N}_0$ ▷ $\mathbf{R}_0 = \mathbf{0}$ by construction
  $\mathbf{c} \leftarrow \mathbf{x}^{(s+1)}$
end for
return $\mathbf{x}^{(S)}$

B JULIA-SET FRACTAL NUMERICAL EXPERIMENTS

We generate binary images of the Julia-set defined by:

$$z_{n+1} = z_n^2 + c, \qquad c = -0.7953604080 + 0.1338052794\, i.$$

The training set contains 256 independently sampled nested zoom paths. Each path consists of an initial 512×512 frame followed by ten successive $2\times$ zooms. The model is trained only on scales 1-8. Scales 9 and 10 are reserved for testing its extrapolation capabilities. Training spatial coverage

decreases rapidly with increasing scale. The transformer receives a 96×96 input patch, while the loss is computed only over its central 64×64 region, providing a 16-pixel context halo on every side. At inference, an entire 512×512 frame is reconstructed by sliding these 96×96 windows across the image and combining their central predictions. For the result of our model shown here, overlapping predictions and test-time augmentations are averaged before producing the final binary image. Comparisons with other models (nearest neighbors and deterministic) for autoregressive rollouts are shown in Figure 6 and Table 2. The deterministic transformer uses the same architecture but is trained directly on coarse-fine image pairs using binary cross-entropy.

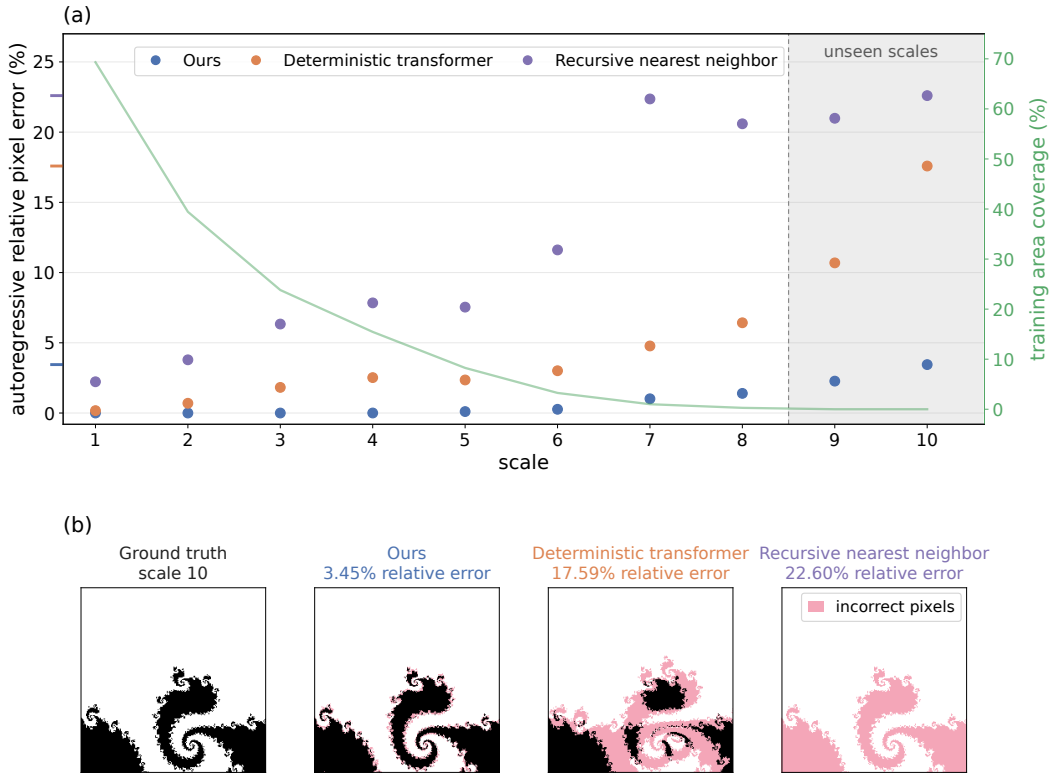

Figure 6: **(a)** Reconstruction relative error across successive refinement levels. Each point shows the error over one of the held-out 512×512 paths from the same Julia-set. The comparison measures how errors accumulate during recursive generation, including at scales beyond those used for training. **(b)** Samples at the final refinement level.

C CIFAR-10 ABLATIONS

All models use a cosine noise schedule with 1,024 discrete observation times. We train each configuration for two million optimization steps and evaluate the effect of count quantization on generation quality. Our intuition was that quantizing the intensity counts would make the refinement problem easier by moving particles in larger discrete packets. We therefore expected quantization to accelerate training and potentially improve generation. In practice, we did not observe either effect.

Table 3 compares raw counts, fixed-divisor quantization, and quantization to a fixed number of bins at each scale. Raw counts achieve the best FID of 21.11, closely followed by division by four with an FID of 23.24. More aggressive quantization substantially degrades generation quality. We therefore use raw counts unless stated otherwise.

Table 2: **Autoregressive rollout error.** Relative error across ten successive spatial scales. We report the mean $\pm$ the standard deviation across all 16 test trajectories. All methods condition autoregressively on their previous prediction. Scales 1–8 are represented during training; scales 9–10 are held out for testing extrapolation capabilities.

Scale	Nearest neighbor	Deterministic	Ours
1	1.934 ± 0.217	0.153 ± 0.017	0.00035 ± 0.00011
2	3.393 ± 0.579	0.633 ± 0.154	0.00127 ± 0.00170
3	4.522 ± 1.476	1.396 ± 0.388	0.00006 ± 0.00014
4	5.491 ± 1.498	1.852 ± 0.514	0.024 ± 0.055
5	8.080 ± 2.098	2.703 ± 0.893	0.310 ± 0.294
6	11.404 ± 2.660	3.601 ± 1.037	0.927 ± 0.451
7	16.831 ± 3.970	5.057 ± 1.611	1.615 ± 0.593
8	24.632 ± 6.016	7.800 ± 2.103	2.435 ± 1.098
9	35.120 ± 10.951	11.887 ± 3.224	3.798 ± 1.897
10	42.636 ± 15.705	18.064 ± 5.704	5.652 ± 2.786

Table 3: Best CIFAR-10 generation results for different count representations. All models are trained for two million optimization steps. FID is computed from 10,000 generated images against the CIFAR-10 test set using 1,024 reverse steps at each of the five refinement levels. We sample from the EMA weights and apply no image post-processing. Lower is better.

Experiment description	FID $\downarrow$
256 bins per scale	47.22
Fixed divisor of 16	62.42
Fixed divisor of 4	23.24
Raw counts	21.11

D Comparison with Discrete Spatial Diffusion

DSD supports both no-flux and periodic boundary conditions, but their computational costs differ substantially. Under no-flux boundaries, the forward transition probabilities are obtained from the exponential of the full Markov generator. For a three-dimensional volume with L^3 voxels, this operator has dimension $L^3 \times L^3$; storing the transition probabilities over T diffusion times therefore scales as $O(TL^6)$. This becomes prohibitive even for moderate 3D domains. For example, a single dense transition matrix for a 256^3 volume contains approximately 2.8×10^{14} entries, corresponding to roughly 2 PB in double precision before accounting for multiple diffusion times. The original DSD paper explicitly notes that the matrix exponential required for no-flux boundaries is expensive.

Periodic boundaries are substantially more tractable because translation invariance makes the transition operator diagonal in Fourier space. DSD can therefore represent the forward process using a single displacement kernel in lieu of a separate transition distribution for every starting voxel. Nevertheless, precomputing this kernel and its cumulative distribution for every diffusion time still requires $O(TL^3)$ memory. At 256^3 voxels and $T = 1500$, storing both arrays in double precision requires approximately 403 Gb. Thus, periodic boundaries remove the L^6 bottleneck, but the original precomputation strategy remains expensive at the scales required for three-dimensional generation.

Table 4 compares the sampling workloads of DSD and recursive refinement under the stated step budgets. DSD operates at full resolution throughout sampling, while our method introduces fine cells progressively. These budgets correspond to approximately 73 times fewer fine-cell updates for our method; this measures spatial workload rather than FLOPs or wall-clock speed.

Table 4: CIFAR-10 sampling comparison, assuming 100,000 reverse steps for DSD and 1,024 steps per scale for our method. Fine-cell updates count the spatial cells processed across all reverse steps; they do not measure FLOPs or wall-clock time.

	DSD (Santos et al., 2026)	Ours (streaming)
Number of parameters	73.9M	37.9M
Fine-grid sizes	32^2 throughout	$2^2, 4^2, 8^2, 16^2, 32^2$
Total reverse steps	$\approx 100{,}000$	5,120
Fine-cell updates	102.4×10^6	1.40×10^6
Particle redistribution	Spatial neighborhood	Four fine cells per coarse cell
FID ↓	46.3	21.1

E FFHQ

Figure 7 shows randomly generated FFHQ samples. Using 50,000 generated images, the model achieves an FID of 39.1 against the training set.

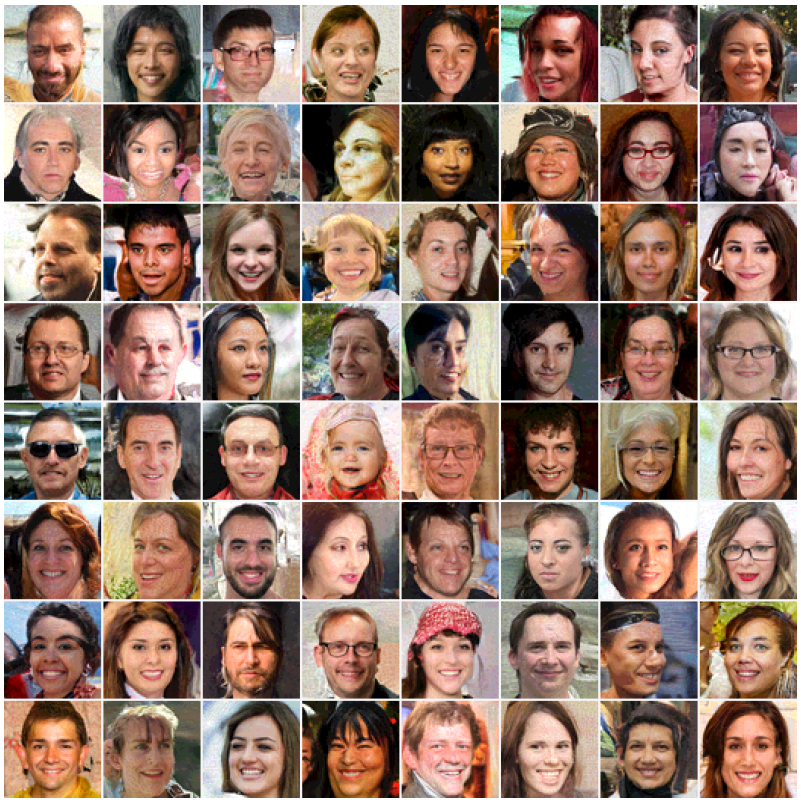

Figure 7: Random FFHQ samples generated with our trained model.

F Refinement from coarse images

We apply the pretrained FFHQ model from Section E to refine 16×16 inputs into 64×64 images without additional fine-tuning. The coarse input supplies the spatial organization and color information, while the model samples the unresolved fine detail. Figure 8 illustrates this capability using portraits featured in an online discussion of PULSE (DeepLearning.AI, 2020). We independently pool the source photographs to construct our inputs and include the published outputs as qualitative context.

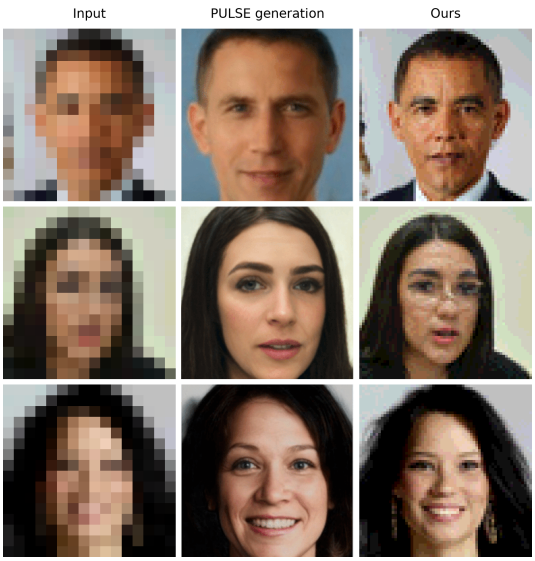

Figure 8: Refinement from coarse facial images. Left: 16×16 inputs obtained by independently pooling the source photographs. Center: published PULSE outputs reproduced from DeepLearning.AI (2020). Right: 64×64 samples generated by our pretrained model conditioned on the inputs shown. Our pooled inputs differ from those used for the published examples, so the figure illustrates model behavior rather than providing a controlled comparison.

G ZERO-SHOT IMAGE SUPER-RESOLUTION

A CIFAR-10 test image can be used directly as the coarse state of our refinement process to test whether the refinement rule learned only at CIFAR-10 resolutions extrapolates beyond its training range. Starting from 32×32 test images, we recursively apply the model's final-scale refinement rule to generate samples at 64×64 and 128×128, without fine-tuning or exposure to images at either target resolution. The resulting samples preserve the coarse semantic content, color structure, and object layout while introducing plausible fine-scale variation and sharper boundaries. We use 64 reverse steps per refinement stage (128 total), with generation taking only a few seconds per sample on a macbook.

This experiment is not intended as conventional supervised super-resolution, since no corresponding high-resolution targets exist for CIFAR-10. Rather, it provides qualitative evidence that the learned local refinement dynamics can be composed recursively and remain useful at spatial scales not encountered during training. Representative samples are shown in Figure 9.

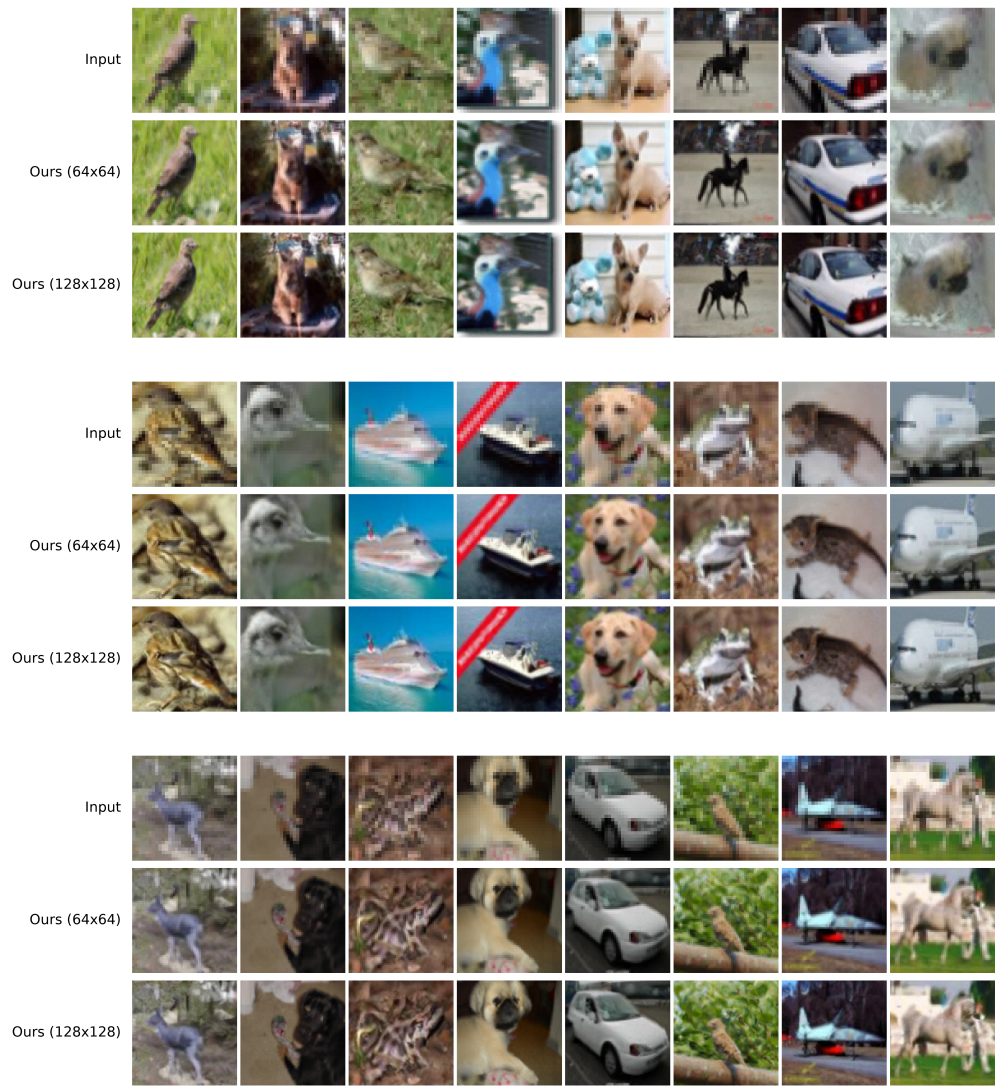

Figure 9: Recursive zero-shot scale extrapolation on CIFAR-10. Starting from 32×32 test images, the model is applied recursively to generate 64×64 and 128×128 samples, despite not being trained at these resolutions. No post-processing is applied.

H AI DISCLOSURE

AI tools were used to identify typos and inconsistencies, verify that all figures and tables were properly referenced in the text, and assist with complex LaTeX code debugging.